uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías Industriales

(2016-2017)

*Trabajo Fin de Grado*

# Reconocimiento y conteo de cubos para automatización de evaluación de destreza manual

---

José Manuel Bueno Carmona

Tutor: Edwin Daniel Oña Simbaña

Leganés, Octubre 2017

# Índice

# Índice de figuras

# Índice de tablas

# Resumen

El ejercicio "*Box and Blocks Test*" para la evaluación de la destreza manual gruesa es una prueba que se ha intentado automatizar en múltiples ocasiones. Aunque resulta sencillo realizar la propuesta con varios sensores, la utilización de una cámara de captación de vídeo para controlar del ejercicio es una opción más económica, aunque no tan exacta. Es por ello que comprobando y combinando los distintos algoritmos de la visión artificial la aplicación puede acercarse a una solución mucho más óptima. Este proyecto se centrará en las ventajas y desventajas que proporciona el método conocido como filtro de partículas al implementarse, junto con otros algoritmos, en el procesamiento de las imágenes durante la realización de las pruebas.

# Abstract

The Box and Blocks Test for the evaluation of gross manual dexterity is an exercise that has been tried to be automatized multiple times. Although it is simple to achieve the proposal with some sensors, the use of a video camera for controlling the test is a more economical choice, but it is not as accurate. Because of this, checking and combining the different algorithms of artificial vision, the aplication can be approached to a more better solution. This project will focus in the advanteges and disadvanteges of the method known as particle filter when it is implemented, with another algorithms, in the images' processing during the realization of the test.

# Introducción

Los avances en medicina de la época actual han incrementado notoriamente la esperanza de vida de los seres humanos, pero con una sociedad más envejecida, y por lo tanto más frágil, es también mayor el número de pacientes que necesitan un plan de rehabilitación personalizado para prevenir futuros problemas de salud. En la mayoría de ejercicios de rehabilitación normalizados, el paciente necesita estar al cuidado de un responsable, limitando esto la atención del mismo a una única persona. Es por ello que crear aplicaciones que permitan a los pacientes realizar los ejercicios por su cuenta, siempre y cuando no exista riesgo de lesión al realizarlo, otorga al equipo médico cualificado la posibilidad de monitorear la rehabilitación de varios pacientes al mismo tiempo.

Este proyecto llama la atención debido a la repercusión que podría tener en el campo de la salud, ya que las aplicaciones médicas de la ingeniería no son muy conocidas en la sociedad. Sin embargo, la creciente necesidad de automatización de la vida cotidiana está haciendo que el ámbito de la ingeniería en general esté en auge incluso en centros sanitarios de toda índole. En especial, lo que se va a tratar en este proyecto está destinado en particular a los centros de rehabilitación.

La robótica asistencial, un campo de la ingeniería, no es vista a lo largo de los años de grado, y es por ello que la profunda investigación que será necesaria para la realización de este trabajo puede abrir la mente hacia otras ramas de la ingeniría más desconocidas.

Durante el proceso de investigación, serán analizados diferentes tipos de programas de seguimiento del movimiento, y con esta información serán comparados en busca del más ideal para llevar a cabo el proyecto en las mejores condiciones posibles.

# Objetivos

El objetivo general de este proyecto comprende la automatización del conteo de cubos en un test de evaluación de destreza manual, utilizando algoritmos dentro del campo de la visión artificial, como un filtro de partículas.

Los objetivos específicos del proyecto son los siguientes:

1. Conocer el funcionamiento del filtro de partículas.
2. Desarrollar la automatización del "*Box and Blocks Test*" (BBT) mediante un filtro de partículas.
3. Destacar las ventajas e inconvenientes de la implementación del filtro de partículas sobre la automatización del BBT.
4. Comprobar los resultados en diferenciación de colores en distintos espacios de color.

# Herramientas

## Materiales del Box and Blocks Test

La prueba conocida como "*Box and Blocks Test*" (BBT) es un método de evaluación de la destreza manual gruesa en pacientes que sufren o han sufrido algún accidente o enfermedad neurológica y, como consecuencia, han perdido movilidad en el brazo. El método consiste en el desplazamiento de una serie de cubos usando una única mano (Greenan, Buxton, Sillo, y Thomas, s.f.).

El ejercicio se realiza con los siguientes materiales (Mathiowetz, Federman, y Wiemer, 1985):

- Una caja de madera.
- Alrededor de 150 cubos de madera de pequeñas dimensiones.
- Una tabla delimitadora que divida ambas partes de la caja.
- Temporizador.

Para preparar la prueba correctamente, en la mitad de la caja, una vez esté abierta, se debe colocar la tabla delimitadora para separarla en dos zonas, y todos los cubos tienen que encontrarse en una de las mismas. El paciente debe encontrarse sentado frente a la caja de manera que los cubos se encuentren en el lado correspondiente a la mano que va a utilizar en el ejercicio. Una vez esté todo preparado podrá dar comienzo a la prueba. El paciente deberá transportar, de uno en uno, el mayor número de cubos posibles de un lado a otro de la caja durante un minuto de tiempo. La puntuación final del ejercicio corresponderá al número total de cubos en la caja (Mathiowetz et al., 1985).

Existen ciertas excepciones a la hora de valorar la puntuación, pues transportar 2 o más cubos en un solo viaje se considerará como un único punto. El movimiento que se debe realizar para que el cubo cuente como un punto debe cruzar por encima la tabla

delimitadora, por lo tanto lanzarlos en vez de transportarlos no será válido (Mathiowetz et al., 1985).

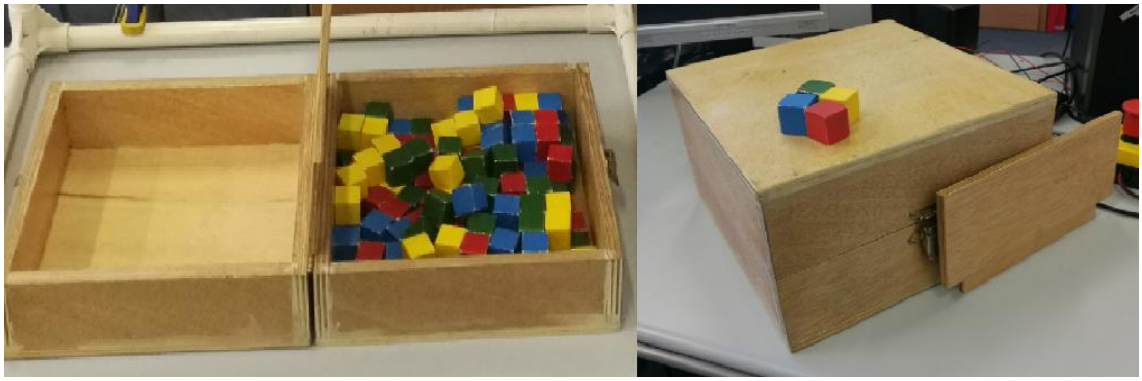


*Figura 1*. Material necesario del BBT: Cubos, caja y tabla delimitadora

**Kinect 2.0**

La cámara Kinect tiene su origen en un proyecto de Microsoft Research conocido como "*El Proyecto Natal*". Sus integrantes buscaban una mejora del entretenimiento en el hogar con la creación de hardware especializado para videojuegos en los que el propio cuerpo humano se convertía en el mando. Lo que no esperaban es que tras la creación de la cámara Kinect, ésta se convirtiera en la herramienta principal para la creación de variadas aplicaciones que afectan a ámbitos como la sanidad, la educación o el turismo. Esta ola de creatividad y mejoras ha sido denominada "*El Efecto Kinect*" (Xataka Windows, 2013).


*Figura 2.* Cámara Kinect 1.0 *(Xataka Windows, 2013)*

El primer producto que se presentó, la cámara Kinect 1.0 para Xbox 360, fue un éxito de mercado. Está compuesta por los elementos necesarios para hacer captura de imágenes en 3D de personas y objetos en movimiento: una cámara, un proyector infrarrojo y micrófonos para la captación de voz. (Xataka Windows, 2013).

*Figura 3.* Cámara Kinect 2.0 *(Xataka Windows, 2013)*

Como nos indica el artículo de Xataka Windows (2013) la cámara Kinect 2.0 superó todas y cada una de las cualidades de su predecesora. La principal diferencia es la incorporación de una cámara "*Time-Of-Flight"* (TOF) de alta resolución, que triplica la fidelidad de reproducción con respecto a la versión 1.0. La detección de personas y objetos mejora hasta ser capaz de reconocer hasta seis personas a la vez, dos más en comparación con la cámara anterior. Estas mejoras son posibles gracias al incremento del alcance de la cámara y a los nuevos sensores infrarrojos especiales para zonas de

menor iluminación. Toda la información es recogida por la cámara gracias a la mejora del procesador que premite leer hasta 2 gigabits de datos del entorno por segundo.

Tabla 1.

Comparación de las carácterísticas de las cámaras Kinect *(Ashley, 2014)*

| Feature | Kinect for Windows 1 | Kinect for Windows 2 |
|---|---|---|
| Color Camera | 640 x 480 @30 fps | 1920 x 1080 @30 fps |
| Depth Camera | 320 x 240 | 512 x 424 |
| Max Depth Distance | ~4.5 M | 8 M |
| Min Depth Distance | 40 cm in near mode | 50 cm |
| Depth Horizontal Field of View | 57 degrees | 70 degrees |
| Depth Vertical Field of View | 43 degrees | 60 degrees |
| Tilt Motor | yes | no |
| Skeleton Joints Defined | 20 joints | 25 joints |
| Full Skeletons Tracked | 2 | 6 |
| USB Standard | 2.0 | 3.0 |
| Supported OS | Win 7, Win 8 | Win 8 |

Para poder utilizar la cámara Kinect en un equipo Windows serán necesarios un adaptador con entrada USB3.0 para conectar el equipo al ordenador y la librería de Kinect SDK para Windows, necesaria para realizar la captura y procesamiento de las imágenes.

**Librería de visión artificial OpenCV**

La biblioteca abierta OpenCV se trata de un proyecto de la compañía Intel que duró desde 1998 hasta principios del nuevo milenio, y que actualmente se puede encontrar gratuitamente en internet. Su objetivo es la ayuda en la resolución de problemas sobre la visión por computador, otorgando herramientas como pequeñas funciones de procesado

de imágenes o algoritmos de complejidad elevada (Kari, Baksheev, Kornyakov, y Eruhimov, 2012).

A día de hoy la librería contiene más de 2500 algoritmos usados en los distintos ámbitos de la visión por computador, la comunidad de usuarios supera los 47000 perfiles y el número de descargas desde la página web es superior a 14 millones. El uso de esta librería se ha implementado en diversas compañías y grupos de investigación. Cabe destacar que las funciones incluidas en OpenCV se pueden aplicar en diversos lenguajes de programación (C, C++, Python, etc.), y también en múltiples plataformas. Tanto es así que se pueden programar de manera sencilla aplicaciones para móviles y tabletas en sistemas operativos como son Android e IOS (OpenCV, s.f.).

# Estado del arte: Seguimiento de objetos

Como nos indican Emilio Maggio y Andrea Cavallaro (2011), la velocidad y la calidad con la que se recogen vídeos en la época actual permiten que en los distintos campos de la tecnología se utilicen con más normalidad las funciones de la visión por computador. Gracias a esto, ahora se pueden crear aplicaciones y funciones que antes no se podían contemplar. Campos como la robótica y los sistemas de vigilancia son los principales ejemplos de sectores que se benefician del crecimiento de estas aplicaciones. Dentro de todas las funciones que se pueden procesar en un vídeo, la localización de un objeto, y en ocasiones su posterior seguimiento, es una herramienta clave para gran parte de las aplicaciones.

En una aplicación de detección, la máquina intentará determinar dónde se encuentra nuestro objeto de interés. Entendemos por objeto de interés a la cosa, cuerpo o parte del cuerpo que necesitamos distinguir en el vídeo (Maggio y Cavallaro, 2011). De esta manera, el objeto de interés en un videojuego pueden ser las manos, mientras que en un sistema de control de tráfico buscamos los coches que circulan por la vía. Para definir

nuestro objeto de interés debemos ser capaces de expresar correctamente las cualidades que tiene, como pueden ser el color o la forma.

# Características de un objeto de interés

Aunque existen diversas propiedades que se pueden calcular del objeto de interés en una imagen, un algoritmo perfecto es complicado de conseguir. Esto es debido a que se necesita que sea eficiente a la hora de hacer la diferenciación, y además robusto frente a cualquier tipo de cambio (por ejemplo, la iluminación), pero ante todo se busca que no tenga una gran capacidad computacional para poder usarlo en aplicaciones a tiempo real. Muchos de los algoritmos existentes no serán capaces de cumplimentar todas las necesidades comentadas, por eso hay que saber elegir qué características y qué métodos usar en cada caso.

## Color

El desarrollo de las imágenes a color no siempre ha sido importante, pues ha ido creciendo a la vez que los elementos de hardware han facilitado la captura de las mismas. La captación del color es distinta dependiendo de los sensores usados, creándose así distintos espacios de color que se eligen dependiendo de la aplicación final necesaria. Se entiende como espacio de color "un método por el que se pueda especificar, crear o visualizar cualquier color." (de la Escalera, 2001).

### *Espacio RGB*

Entre los espacios de color existentes el más intuitivo es el espacio RGB (*"Red Green Blue"*). Se basa en captar por separado de los tres colores básicos; rojo, azul y verde. Las cámaras de captación usan filtros RGB para crear las imágenes, diferenciando cada color por su longitud de onda usando prismas o espejos dicroicos. El espacio RGB tiene el inconveniente de mezclar en sus componentes las variables de

saturación, tono e intensidad del color, lo que puede ocasionar dificultades a la hora de modificar la imagen (de la Escalera, 2001).

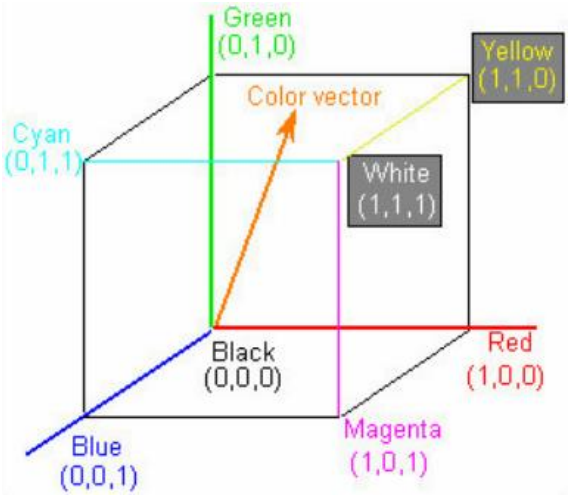



*Figura 4.* Representación gráfica del espacio de color RGB *(González Cid, s.f.).*

***Espacios HSI y HSV***

Existen espacios que tratan de percibir las imágenes de la misma manera que el cerebro humano. El espacio HSI (*"Hue Saturation Intensity"*) es un ejemplo de este tipo de espacios. Maneja los siguientes componentes: tono, saturación o cromatismo y brillo. Estos parámetros no son tan intuitivos como los colores del espacio RGB. Para comprender mejor de lo que estamos hablando, de la Escalera (2001) explica estos parámetros de la siguiente manera:

- Brillo: sensación que indica si un área está más o menos iluminada.
- Tono: sensación que indica si un área parece similar al rojo, amarillo, verde o azul o a una proporción de dos de ellos. (…)
- Croma: La coloridad de un área respecto al brillo de un blanco de referencia.
- Saturación: La relación ente la coloridad y el brillo.

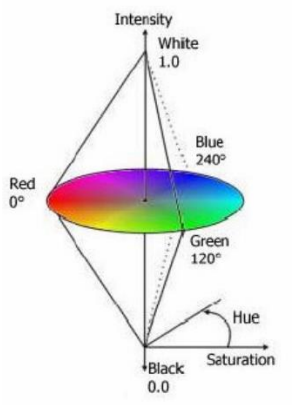



*Figura 5.* Representación gráfica del espacio de color HSI *(González Cid, s.f.)*

Para diversas aplicaciones se han ido generando variaciones de este mismo espacio de color, del cual se destaca el espacio HSV ("*Hue Saturation Value*"), donde la componente "V", de valor, se define como brillo o mate. Aunque es muy parecido al modelo HSI, se diferencia en que un valor alto del tercer componente en el primer espacio significa que tenemos un color muy brillante, mientras que en el espacio HSV un valor máximo de mate significa el color blanco, independientemente de los valores de H y S (González Cid, s.f.).

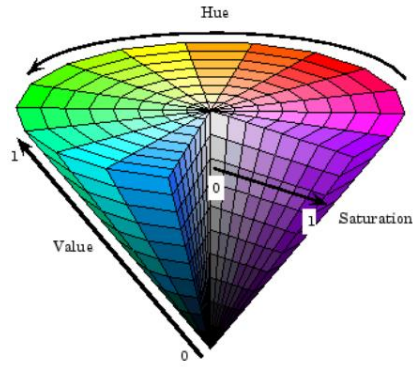



*Figura 6.* Representación gráfica del espacio de color HSV *(González Cid, s.f.)*

***Espacio CIELAB***

La organización conocida por su título francés Commission Internationale de l'Éclairage (CIE) ha definido unos espacios de color para expresarlo de manera más objetiva, entre ellos el espacio L*a*b también conocido como CIELAB. Este método

correlaciona los valores numéricos del color, lo cual convierte este espacio en uno de los más uniformes, y por lo tanto también en uno de los más populares. Este espacio de color se basa en el hecho de que un objeto no puede ser rojo y verde a la vez. Se puede diferenciar el color diciendo si el objeto es más rojo y menos verde ó más verde y menos rojo, determinado por la variable "a". De la misma manera, la variable "b" se refiere a la comparación de los colores amarillo y azul. Por último, "L" es la luminosidad. Este espacio de color es tremendamente útil a la hora de comparar colores (Konica Minolta, s.f.).

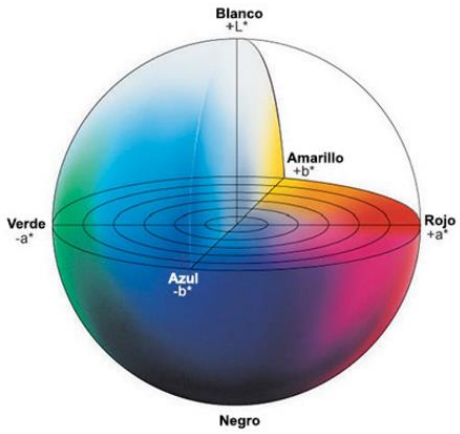



*Figura 7.* Representación gráfica del espacio de color L*a*b *(Konica Minolta, s.f.)*

## Bordes

Comprobando la variación de intensidad que se produce en una imagen es posible comprender la información sobre los límites y bordes de los objetos que aparecen en la misma. Los métodos de detección de bordes se basan en los operadores derivadas. Si se aplica la primera derivada, el gradiente, se comprobará los grandes cambios de intensidad. En cambio, mediante el uso de la segunda derivada, la laplaciana, se buscan los pasos de respuesta positiva a negativa o viceversa (de la Escalera, 2001).

Como no se puede suponer que el valor de gris en un objeto sea constante, se necesita realizar una fase en la que se distingan los bordes correctos y el ruido (de la Escalera, 2001).

**Texturas**

Es posible detectar el aspecto homogéneo que presenta una misma textura en una imagen, independientemente de que el valor de los píxeles en la zona homogénea sea o no igual. De esta manera, se definen las diferentes texturas encontradas como los diferentes patrones detectados. Es posible buscar los patrones de la imagen de dos maneras: la primera, analizando los valores estadísticos de los niveles de gris y picos en un área, se denomina aproximación estadística, y la segunda, conocida como aproximación frecuencial y que compara los valores altos y picos en la transformada de Fourier con la frecuencia de repetición (de la Escalera, 2001).

*Figura 8.* Muestra de distintas texturas

# Representación de objetos

Cuando se ha detectado el objeto de interés, es importante representarlo en la imagen de la manera correcta, de manera que otorgue la información necesaria. Por eso, si se está buscando una única pelota en la imagen, será sencillo representarlo con un punto,

mientras que si se quieren detectar los gestos faciales de una persona con una imagen de su cara se necesitará dibujar, de la forma más posible, todas las líneas expresivas que pueden afectar la detección del gesto.

Las representaciones deben ser lo más acertadas posibles, pero no siempre se puede conseguir que la representación encaje perfectamente con el objeto de interés. Los modelos de representación se pueden definir de manera previa o instantáneamente en cada fotograma. Podemos diferenciar los métodos de representación en tres grupos: básicos, articulados y deformables. (Maggio y Cavallaro, 2011).

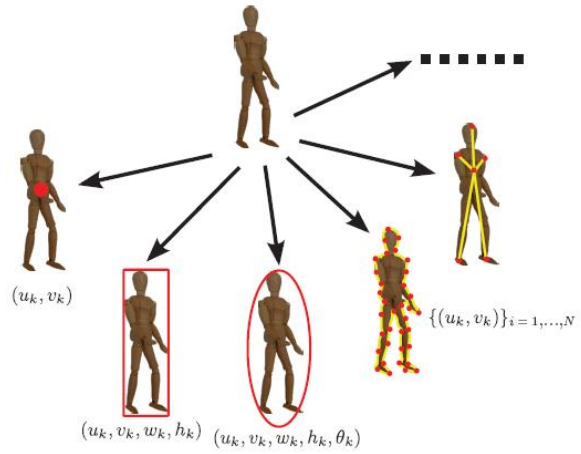



*Figura 9.* Modelos de representación de objetos de interés (Maggio y Cavallaro, 2011)

**Modelos básicos**

Maggio y Cavallaro (2011) definen tres tipos de módelos de representación básicos: aproximación a un punto, al área y al volumen.

La manera más simple de determinar un objeto es con un punto. En estos casos, se usa un único punto para determinar la posición del objeto de interés. No obstante, como es una representación que no tiene en cuenta el tamaño o forma de los objetos de interés, no se podría detectar bien su posición si el objeto se encuentra ocluido (Maggio y Cavallaro, 2011).

También se puede encerrar el objeto en un área, como por ejemplo un rectángulo o una elipse. Los datos de la forma que se busca pueden venir predefinidos desde un primer momento o pueden usar los datos recabados de la imagen para definirse en cada instante (Maggio y Cavallaro, 2011).

Por último, cuando se necesita una representación 3D del objeto, es mucho más útil representar el objeto dentro de un volumen. Se puede aplicar en casos en los que se tiene acceso a varias cámaras con planos solapados que compartan información. Para poder aplicarlo en visión monocular se tienen que definir precisa y correctamente las suposiciones que se utilizan para valorar el movimiento del objeto. En el caso de que se tengan los datos de volumen del objeto bien definido y éste no sea deformable, representar el objetivo volumétricamente será más sencillo (Maggio y Cavallaro, 2011).

**Modelos articulados**

Los modelos articulados utilizan otros tipos de representación más básicos para encerrar cada una de las partes del cuerpo de interés. Además, las partes estarán relacionadas entre sí con conexiones topológicas y restricciones del movimiento que harán del cuerpo una cadena de articulaciones cinemáticas. (Maggio y Cavallaro, 2011).

Estos modelos se usan principalmente para realizar algoritmos de captura de movimiento de personas o animales. Por ejemplo, en la industria cinematográfica, es muy utilizado para crear personajes de animación con movimientos más reales. Cuando queremos capturar el movimiento de un cuerpo tenemos dos opciones. Los métodos utilizados para conseguir estas representaciones se nombran y comparan en el artículo de Ceseracciu, Sawacha y Cobellyi (2014): las técnicas "*marked-based*" y "*markerless*" para la captura de movimiento de los cuerpos articulados.

**Modelos deformables**

Maggio y Cavallaro (2011) introducen el último tipo de represetación para los casos en los que no se tienen suficientes datos sobre la forma y el tamaño de los objetos de interés o estas características varían y se deforman a lo largo de la búsqueda. En estas ocasiones se pueden usar los siguientes métodos: modelo fluido, búsqueda de contornos y modelo de distribución de puntos.

El modelo fluido trata de detectar zonas homogéneas o puntos específicos en vez del objeto completo, como por ejemplo la búsqueda de esquinas. En casos en los que los objetos a identificar estén cerca el uno del otro, el modelo no sabrá diferenciar qué puntos o qué zonas de interés pertenecen a cada objeto (Maggio y Cavallaro, 2011).

También se puede detectar el contorno de objetos que tengan una misma característica, y posteriormente dibujar los puntos que pertenecen al contorno para remarcar la silueta de nuestro objeto de interés (Maggio y Cavallaro, 2011).

En casos en los que se tenga la información suficiente de cómo se deforma el objeto a detectar, podemos aplicar el modelo de distribución de puntos, que intentará separar las zonas deformables en triángulos para poder entender mejor cada deformación. Esta es la tecnología usada para la detección y comprensión de gestos faciales (Maggio y Cavallaro, 2011).

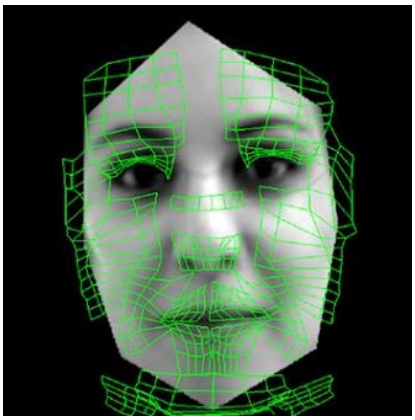


*Figura 10.* Ejemplo de software de detección de gestos faciales *(Monge, 2011)*

# Segmentación

Diferenciar todos los objetos de una imagen es una tarea compleja que aún tiene muchos años de investigación por delante. No obstante, existen distintas técnicas para hacer una separación de los objetos según alguna de sus características. Los procesos por separado demuestran muchos errores, por eso actualmente se utilizan varios métodos de manera conjunta para realizar una mejor detección (Platero, 2017).

De la Escalera (2001) nos enumera las tres propiedades básicas de la segmentación:

- Similitud. Cada uno de los pixeles de un elemento tiene valores parecidos para alguna propiedad.
- Discontinuidad. Los objetos destacan del entorno y tienen por tanto unos bordes definidos.
- Conectividad. Los pixeles pertenecientes al mismo objeto tienen que ser contiguos, es decir, deben estar agrupados.

## Detección de bordes

Platero (2017) destaca que aunque existan métodos de cálculo de bordes en una imagen, los resultados generados no son completamente precisos, pues es necesaria una etapa posterior en la que se definan las fronteras del objetos usando la información recabada.

### *Metodos de detección de bordes con el gradiente*

Una vez se tienen los datos recabados de la detección de bordes por gradiente, Platero (2017) define la norma de conexión de los píxeles de manera que “dos píxeles serán considerados pertenecientes a una misma frontera si presentan alguna condición de conectividad y las diferencias entre sus gradientes no superan un determinado umbral”, lo cual se puede expresar matemáticamente como:

$$|G_1 - G_2| \leq T_M$$

$$|\theta_1 - \theta_2| \leq T_A$$

donde $G_i$ es el módulo del gradiente y $\theta_1$ el argumento del mismo.

***Transformada de Hough***

Aparte de los métodos que utilizan la información de las zonas cercanas de cada pixel, existen métodos que utilizan toda la infomación recabada en la foto para realizar la detección, entre los que destaca la transformada de Hough. El algoritmo intenta detectar las formas geométricas más sencillas para posteriormente evaluar los bordes en el conjunto total de la imagen. Su mayor desventaja es su alto valor computacional (Platero, 2017). La transformada de Hough se puede aplicar para la detección de líneas y circunferencias, aunque existe un tercer algoritmo de este método que detecta las figuras geométricas con formas que no se pueden expresar analíticamente (de la Escalera, 2001).

**Segmentación de regiones homogéneas**

La segmentación por regiones se realiza cuando la característica que estamos evaluando se repite en un área, al contrario que el caso anterior que se valoraba solo sobre la línea de los contornos.

***Técnica de umbralización***

El objetivo de la umbralización es diferenciar en una imagen los elementos con una o varias características, de manera que se eliminan de la fotografía todos los datos que no nos interesa evaluar. El resultado de una umbralización se ofrece normalmente en formato binario, es decir, dos valores que son el blanco y el negro. De esta manera, nuestra imagen umbralizada marcará con uno de los dos colores, por ejemplo el blanco, la parte de la imagen que nos interesa.

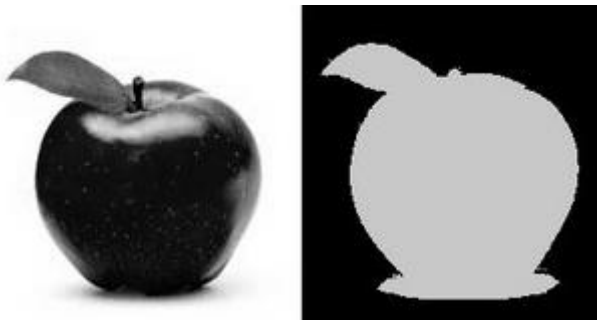


*Figura 11*. Ejemplo de umbralización *(OpenCV, s.f.)*

***Crecimiento de regiones***

Consiste en la elección de un número de píxeles, denominados píxeles semilla, que se compararán con puntos vecinos de la imagen de manera que, si estos tienen la misma característica buscada, se considerarán como pertenecientes a la misma región y se compararán los píxeles vecinos con los nuevos con la misma finalidad. La búsqueda acaba en el momento en que los píxeles de los contornos del área calculada no son iguales (de la Escalera, 2001).

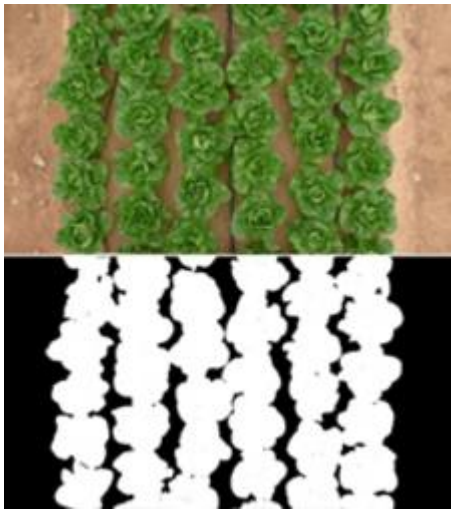


*Figura 12.* Ejemplo de segmentación por crecimiento de regiones (García Mateos, Jaén Terrones, Escarabajal Henarejos, Molina Martinez, y Hernández Hernández)

# Detección de movimiento

Muchas de las aplicaciones de detección de objetos y conteo utilizan programas que comparan dos fotogramas del vídeo usado para distinguir los cuerpos que se han desplazado a lo largo del tiempo. Estas aplicaciones son sencillas y muy útiles, sobretodo en programas de vigilancia. En este apartado se explicarán los distintos métodos para la creación de este tipo de aplicaciones con la comparación de imágenes.

Los movimientos del cuerpo humano se pueden capturar usando las técnicas denominadas "*marked-based*" y "*markerless*", que son conocidas por su uso en películas de animación y los videojuegos.

**Comparación de fotogramas**

El método de comparación de fotogramas utiliza la imagen en el tiempo actual y otra foto objetivo para comprobar los cambios que han surgido entre ellos. Este fotograma objetivo puede ser una imagen inicial del entorno o el fotograma anterior. Arturo de la Escarlera (2001) nos indica que la imagen resultante es una imagen binaria, cuyo valor positivo indica una diferencia en la escala de grises mayor a un valor de sensibilidad elegido previamente:

$$p(x,y)_{resultante} = \begin{cases} 1 & si\ |p(x,y)_1 - P(x,y)_2| \geq sensibilidad \\ 0 & Cualquier\ otro\ caso \end{cases}$$

En caso de que estemos comprobando dos fotogramas seguidos, claramente aparecerá un valor positivo cuando mueve un cuerpo a una posición nueva. No obstante, también aparecerá esta respuesta en la posición anterior del cuerpo. Para evitar el problema se deberá realizar un tratamiento sobre la imagen posterior que una ambas respuestas en una sola figura (Hounslow, 2014).

*Figura 13.* Problema del método de comparación de movimiento fotograma a fotograma *(Hounslow, 2014)*

Como el ordenador entiende como movimiento la variación del nivel de gris de cada pixel entre ambas imágenes, el resultado final no solo mostrará el movimiento de los objetos. Variaciones en la luminosidad o movimientos de la propia cámara generan también respuestas positivas en el resultado (de la Escalera, 2001).

**Métodos Marked-Based y Markerless**

Suzzanne Rebello (2009) explica que los métodos más usados en la industria del cine y los videojuegos para traspasar los movimientos de una persona a una imagen por ordenador pueden parecer primitivos. En su artículo destaca el método "*marked-based*", que detecta las uniones de las articulaciones con distintos puntos o marcas, que se pueden predefinir por ordenador o que pueden ser marcas físicas en el cuerpo del actor que sean visible para la cámara.

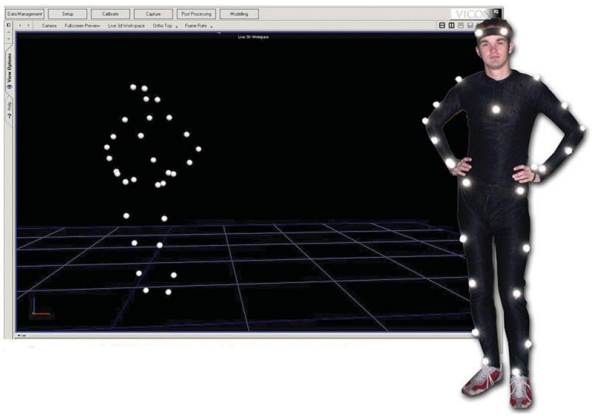


*Figura 14. Ejemplo de detección de marcas físicas de un traje optoelectrónico mediante cámaras (Rebello, 2009)*

Aunque en los últimos años se está introduciendo en estas aplicaciones la captura del movimiento sin necesidad de marcas, el método "*markerless*". Cesseracciu et al. (2014) hacen una comparación entre estos dos métodos en el campo clínico. En conclusión, el ajuste de los parámetros basándonos en la anatomía humana hace las aplicaciones clínicas posibles sin aplicar marcas, pero pierde precisión a la hora de valorar la rotación de las articulaciones. El punto fuerte del método sin marcas es que ahorra tiempo de preparación de los pacientes en cada uso, pues no será necesario definir los puntos por ordenador o poner marcas físicas en el cuerpo.

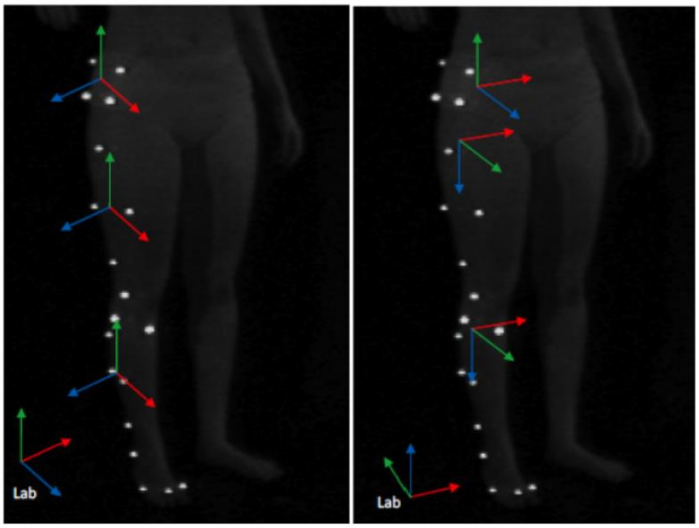



*Figura 15. Referencias creadas por los métodos "marked-based" (Izquierda) y "markerless" (derecha) (Ceseracciu et al., 2014).*

## Detección por aprendizaje

Es posible entrenar a una máquina para que sea capaz de reconocer un objeto de interés después de haber recibido una gran cantidad de fotografías en las cuales aparecen ejemplos del objeto en cuestión. Es un intento de reproducir el mismo trabajo que realiza el cerebro humano, que es capaz de reconocer las cosas que le rodean más fácilmente porque ha tenido la opción de ver ejemplos anteriores de las mismas. Incluso si no se ha visto antes, podemos relacionar algo nuevo con objetos de la misma clase.

Por ejemplo, el ser humano es capaz de reconocer que un libro es un libro sin importar el color de su portada, el tamaño o si es un libro que nunca antes hemos visto.

Este tipo de métodos requiere de una cantidad grande de ejemplos y en ellos se debe marcar dónde se encuentra el objeto de interés. Una vez el programa tenga la batería de fotografías correspondiente, comparará todas las imágenes para detectar qué características en común tienen todas las zonas márcadas en la imagen. Posteriormente, en la fase de detección, el programa extraerá las características resultantes a lo largo de las nuevas imágenes que reciba.

Es posible diferenciar entre dos tipos de ejemplos; los que contienen el objeto de interés son considerados positivos, mientras que aquellos que no lo contienen se consideran negativos. Es importante que los ejemplos positivos que se otorguen a la máquina sean diversos tanto en cuanto a variables como a posición, fondo o iluminación, pues cuanto más variables sean más posibilidades habrá de que se reconozcan en cualquier condición. Por otra parte, los negativos previenen las confusiones en la búsqueda. Es decir, si se planea utilizar la mano para agarrar el objeto buscado, será útil añadir diversas fotos de manos como negativos para que a la hora de dibujar la posición del objeto no se confunda con los dedos o la mano completa.

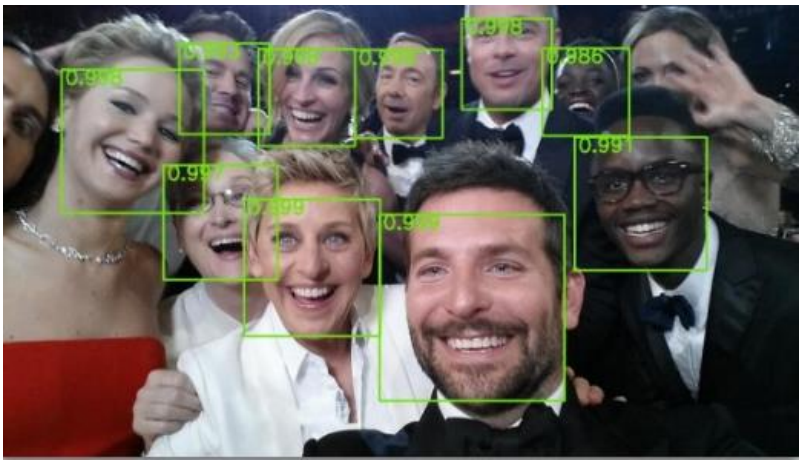


*Figura 16.* Ejemplo de detección de caras comn método de aprendizaje profundo *(Vaas, 2015)*

# Filtro bayesiano

El teorema de Bayes es una forma de comprobar la probabilidad de un suceso usando la información que había de lo ocurrido anteriormente. Matemáticamente se expresa como (Maggio y Cavallaro, 2011):

$$P(A/B) = \frac{P(A) * P(B/A)}{P(B)}$$

Para entender mejor cada elemento de la probabilidad de Bayes, se podrían expresar las probabilidades como:

$$Posteriori = \frac{Priori * Condicional}{Total}$$

Los filtros bayesianos de seguimiento según Maggio y Cavallaro (2011), tienen como característica que son recursivos, es decir, que no necesitan un historial de los datos a lo largo del tiempo porque para determinar la posición del objeto en el momento actual $x_t$, solo es necesaria la predicción de la posición en el instante anterior $x_{t-1}$ y la observación del instante actual $z_t$. La recursión queda determinada por $g_t$, la ecuación de observación:

$$z_t = g_t(x_t, n_t) \quad (1)$$

y la ecuación de estado de la dinámica, $f_t$:

$$x_t = f_t(x_{t-1}, m_{t-1}) \quad (2)$$

donde $\{n_t\}_{t=1,\dots}$ y $\{m_t\}_{t=1,\dots}$ es el posible error producido por el ruido.

El objetivo del filtro de Bayes (Maggio y Cavallaro, 2011) es la estimación de la función de densidad de probabilidad del objeto en el momento t dados los datos de

observación a lo largo del tiempo: $P_{t|t}(x_t|z_{1:t})$. Para ello realizaremos una estimación recursiva en dos pasos:

- Predicción: con los datos de la dinámica y la función de probabilidad calculada en el momento anterior, se halla la función de densidad actual con la ecuación de Chapman-Kolmogorov:

$$P_{t|t-1}(x_t|z_{1:t-1}) = \int f_{t|t-1}(x_t|x_{t-1}) P_{t-1|t-1}(x_{t-1}|z_{1:t-1}) dx_{t-1} \quad (3)$$

- Estimación: cuando se obtiene la nueva observación, se usa la regla de Bayes:

$$P_{t|t}(x_t|z_{1:t}) = \frac{g_t(z_t|x_t) P_{t|t-1}(x_t|z_{1:t-1})}{\int g_t(z_t|x_t) P_{t|t-1}(x_t|z_{1:t-1}) dx_t} \quad (4)$$

Tanto el filtro de Kalman como el filtro de partículas se basan en el teorema de Bayes.

# Filtro de Kalman

En el escrito "*A New Approach to Linear Filtering and Prediction Problems*" publicado en el año 1960, el Dr. Kalman presenta una solución particularmente útil para la estimación del estado dinámico de objetos en el espacio. Esta solución, conocida más comúnmente como filtro de Kalman, elimina los requerimientos estacionarios del método que se usaba anteriormente, el filtro de Weiner. Al comienzo, su principal uso fue en el campo aeroespacial, pues su primera aplicación fue en el proyecto "*Apollo*" de la NASA. No obstante, debido a la simplicidad de las operaciones, no tardó mucho en ser aplicado a otros campos (McGee y Schmidt, 1985).

Maggio y Cavallaro (2011) desarrollan el filtro de Kalman asumiendo que los problemas donde se aplican cumplen linealidad en las ecuaciones de observación (1) y

de estado (2) y que todo el ruido es Gaussiano, considerando entonces el problema óptimo en cuestión del error de mínimos cuadrados en el estado estimado. Asumiendo estas probabilidades, las expresiones lineales de observación y estado tienen la siguiente forma:

$$z_t = G_t x_t + n_t \tag{5}$$

$$x_t = F_t x_{t-1} + m_{t-1} \tag{6}$$

donde las variables $n_t$ y $m_{t-1}$ tienen como media el valor cero y como covarianza los valores $R_t$ y $Q_t$, respectivamente.

La posición medida en el estado anterior, $x_{t-1}$, se puede expresar en base a su media y su covarianza, $\bar{x}_{t-1}$ y $C_{t-1}$, entonces, si se aplica la fórmula de predicción de Bayes (3) y el estado dinámico del problema (6), la ecuación de predicción de la nueva media es:

$$\bar{x}_{t|t-1} = F_t \bar{x}_{t-1}$$

y predicción de la covarianza en el momento actual:

$$P_{t|t-1} = F_t P_{t-1} F'_t + Q_t$$

De la misma manera, es predecible la observación (5):

$$\hat{z}_t = G_t \bar{x}_{t|t-1}$$

Cuando obtenemos la nueva observación desde la ecuación (6), podemos definir con la ecuación de estimación (2) la media residual:

$$\bar{r}_t = z_t - \hat{z}_t$$

y la covarianza residual:

$$S_t = G_t P_{t|t-1} G'_t + R_t$$

Con estos datos se obtiene el dato conocido como ganancia de Kalman:

$$K_t = P_{t|t-1} G'_t S_t^{-1}$$

Finalmente, para completar la recursión, con estos resultados se calculan los datos de media y covarianza en el momento actual:

$$\bar{x}_t = \bar{x}_{t|t-1} K_t \bar{r}_t$$

$$C_t = (I - K_t G_t) C_{t|t-1}$$

Los problemas que no presentan cualidades de linealidad, ruido Gaussiano o ambas, no tendrán resultado óptimo. En estos casos, es evitable la no linealidad del problema con el uso de la aproximación de Taylor de primer orden. Tras esta modificación, se puede aplicar el filtro de Kalman normalmente. Este método es conocido como "*Extended Kalman Filter*" (Turner, 2013).

# Filtro de partículas

El filtro de partículas se remonta a la década de 1940, cuando Nicholas Metropoli decidió realizar un estudio de sistemas con partículas centrándose más en analizar las propiedades del conjunto de las mismas antes que en las partículas individualmente (Díaz Abreu, 2015). Para explicar esto, comparó las investigaciones con el juego de cartas *Solitario*. Ganar el juego es muy complicado, pero un jugador que haga cientos de partidas habrá ganado algunas veces.

Se puede decir que un filtro de partículas es un método de Monte Carlo. Según José Ignacio Illana (2013), se denominan métodos de Monte Carlo a aquellos que "consisten en resolver un problema mediante la invención de juegos de azar cuyo comportamiento simula algún fenómeno real gobernado por una distribución de probabilidad (e.g. un proceso físico) o sirve para realizar un cálculo (e.g. evaluar una integral)".

Este modelo, aun siendo anterior históricamente al filtro de Kalman, no se pudo implementar correctamente hasta la década de 1980 debido a las barreras tecnológicas, pues una de las desventajas de este método es su alto valor computacional. A día de hoy

aún es un problema que influye en las aplicaciones que pretenden usar este algoritmo (Díaz Abreu, 2015).

Dependiendo de la rama de investigación donde se aplica este método, se le ha otorgado nombres distintos. En el campo de la visión por computador es más común el uso de "*Condensation Algorithm*" ó Filtro "*Bootstrap*" (Díaz Abreu, 2015).

**Aplicaciones del fltro de partículas**

En la actualidad, el filtro de partículas es muy importante para los programas de localización de objetos. Además de en el campo de la visión artificial, el algoritmo se utiliza con mucha frecuencia en el campo de la robótica.

El uso del filtro de partículas en el campo de la robótica destaca en la realización del seguimiento de vehículos móviles, como puede ser un dron. Como se puede apreciar en la imagen 17, las partículas, aunque se han creado en una zona alejada del objeto, pueden predecir en qué lugar se encuentra el dron que circula por el pasillo. En cuanto el filtro es capaz de detectar el dron, las partículas comenzarán a moverse prediciendo la trayectoria del vehículo, pudiendo recabar de manera precisa la posición real (Grzonka, Grisetti, y Burgard, 2009).

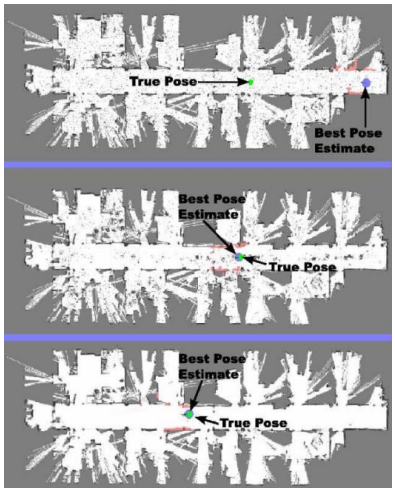



*Figura 17.* Detección de la posicion del dron en un pasillo usando un filtro de partículas (Grzonka et al., 2009)

En sistemas de vigilancia, los filtros de partículas tienen gran importancia para realizar algoritmos de seguimiento robusto frente a oclusiones y que permitan el seguimiento múltiple. Actualmente se están desarrollando aplicaciones capaces de reidentificar a sujetos que han aparecido en cámaras distintas (Vezzani, Baltieri, y Cucchiara, 2015).

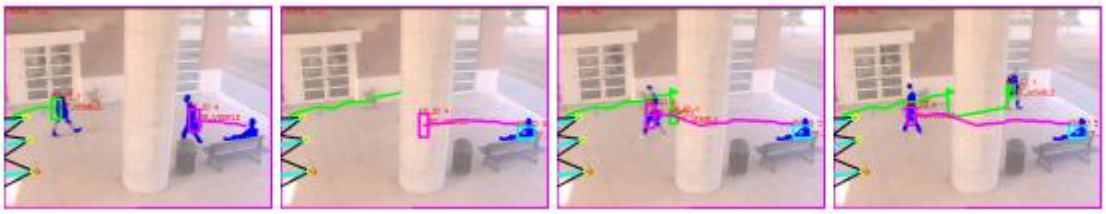


*Figura 18.* Ejemplo de detección de oclusiones usando un filtro de partículas (Vezzani et al., 2015)

**Sequential Implementation Sampling (SIS)**

El filtro de partículas no otorga un resultado exacto, pero sí aproxima a la solución más probable. Se puede aplicar en problemas no lineares con funciones variables en el tiempo (Maggio y Cavallaro, 2011), lo cual supone una mejora respecto al filtro de Kalman. Es un sistema de localización MHL, por lo que en el problema se estarán evaluando distintos casos. Los casos, o partículas, se definirán por dos parámetros: trayectoria y peso: $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{NP}$. La probabilidad de que el objeto de interés siga una trayectoria según los datos observados, $z_{0:t}$, se define como la suma de las trayectorias según su peso (McNames, 2012):

$$P(x_{0:t}|z_{0:t}) \approx \sum_{i=1}^{np} w_t^i \delta(x_{0:t} - x_{0:t}^i)$$

donde la suma de los pesos cumple la condición:

$$1 = \sum_{i=1}^{np} w_t^i \tag{7}$$

Se llama peso a la probabilidad de acierto que tiene cada partícula a lo largo del tiempo. Según Diaz Abreu (2015), los pesos vienen definidos por la distribución objetivo $P\left(x_{0:t}^{i}, z_{0:t}\right)$. Pero la evaluación de esta distribución es tan complicada que en ocasiones es preferible el uso de una segunda distribución a la cual se llamará distribución de importancia, $Q(x_{0:t}^{i}, z_{0:t})$. La distribución de importancia no viene dada por los datos del problema, sino que es elegida por el usuario y en ella será donde se muestreen todos los datos. Tiene como característica que tiene que incluir todo el dominio de la distribución objetivo, entonces se cumple que:

$$P\left(x_{0:t}^{i}, z_{0:t}\right) > 0 \;\rightarrow\; Q\left(x_{0:t}^{i}, z_{0:t}\right) > 0.$$

Aunque se utilice la distribución de importancia para facilitar los cálculos, el muestreo se está realizando en la función equivocada. Para arreglarlo es necesario una corrección de cada partícula de manera que:

$$w_t^i \propto \frac{P\left(x_{0:t}^{i}, z_{0:t}\right)}{Q(x_{0:t}^{i}, z_{0:t})} \tag{8}$$

El problema se basa en el teorema de Bayes, es decir, el algoritmo resultante tiene que ser recursivo y en este caso Maggio y Cavallaro (2011) aprovechan esta característica para hacer uso de las propiedades de las cadenas de Markov, de manera que sea posible expresar la función de importancia como:

$$Q(x_{0:t}, z_{0:t}) = \; Q(x_t|x_{0:t-1}, z_{0:t})Q(x_{0:t-1}, z_{0:t-1})$$

y se asume:

$$Q(x_{0:t}, z_{0:t}) = \; Q\left(x_t\middle|x_{t-1}^{i}, z_t\right)Q(x_{0:t-1}, z_{0:t-1}) \tag{9}$$

Según Turner (2013), la función recursiva de una distribución conjunta se define como:

$$P(x_t|z_{0:t}) = P(z_t|x_t)\int P(x_{t-1}|z_{0:t-1})\frac{P(x_t|x_{t-1})}{P(z_t|z_{0:t-1})}dx_{t-1}$$

McNames (2012) propone la siguiente aproximación a la recursividad de los pesos de cada partícula. Se puede decir que la distribución objetivo depende de:

$$P(x_{0:t}|z_{0:t}) \propto P(z_t|x_t)P(x_t|x_{t-1})P(x_{0:t-1}|z_{0:t-1}) \quad (10)$$

Por lo tanto, usando las ecuaciones, si en nuestra aproximación de pesos (8) sustituimos las ecuaciones de distribución objetivo (10) y distribución de importancia (9), el peso depende de:

$$w_t^i \propto \frac{P\left(z_t|x_t^i\right)P\left(x_t^i|x_{t-1}^i\right)P\left(x_{0:t-1}^i|z_{0:t-1}\right)}{Q\left(x_t|x_{t-1}^i, z_t\right)Q\left(x_{0:t-1}^i|z_{0:t-1}\right)}$$

que es posible simplificar de la siguiente manera:

$$\widetilde{w}_t^i = \frac{P(z_t|x_t)P(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{o:t-1}^i, z_{0:t})} \frac{P\left(x_{0:t-1}^i|z_{0:t-1}\right)}{Q\left(x_{0:t-1}^i|z_{0:t-1}\right)}$$

$$\widetilde{w}_t^i = \frac{P\left(z_t|x_t^i\right)P(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{o:t-1}^i, z_{0:t})} w_{t-1}^i$$

Para acabar, es necesario que los pesos, una vez calculados, cumplan la condición expresada en la ecuación (7). Llegados a este punto, hay realizar una normalización de los pesos:

$$w_t^i = \frac{\widetilde{w}_t^i}{\sum_{j=1}^{np} \widetilde{w}_t^j}$$

Este método de cálculo reiterado que sólo necesita de un momento inicial para empezar es conocido en el campo de la estadística como "*Sequential Importance Sample*" (SIS) .

***Problema de degeneración del SIS***

Cuando el algoritmo SIS se pone en práctica, tras haber realizado varias iteraciones se observa que hay partículas que tienen un peso mucho más significativo que el resto. Por consiguiente, habrá partículas cuyo peso se acerque mucho a 0 y dejen de ser relevantes para el problema que estamos tratando, y se pierda el tiempo calculando

datos que no van a otorgar ningún resultado útil (Turner, 2013). Se puede estimar el grado de degeneración del filtro con el tamaño eficiente de la muestra, que se define como (Orhan, 2012):

$$N_{eff} = \frac{1}{\sum_{i=1}^{np}\left(w_t^i\right)^2}$$

Cuando el valor de $N_{eff}$ es muy pequeño, se considera que el grado de degeneración que existe en el problema es grande.

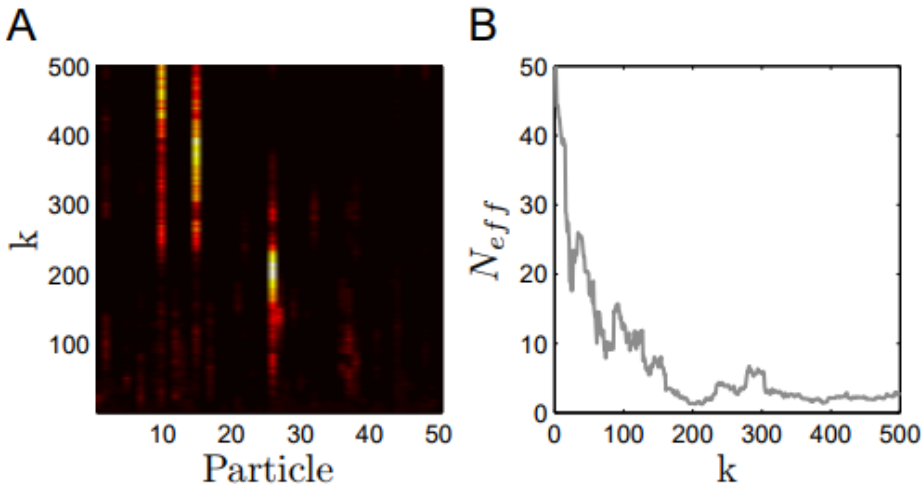



*Figura 19*, (A) Se muestran los pesos de 50 partículas a lo largo del tiempo, k, de manera que los colores más cálidos representan pesos mayores. (B) Tamaño efectivo de la muestra, Neff, a lo largo del tiempo *(Orhan, 2012)*.

Para evitar la degeneración se puede incrementar el número de partículas a evaluar. No obstante, es un método poco práctico que va incrementar el valor computacional. La manera más común de evitar el problema es el uso de técnicas de remuestreo (Turner, 2013).

***Elección de la distribución de importancia***

Turner (2013) explica que la elección correcta de la distribución de importancia es crucial para que el problema de degeneración tenga un efecto más lento sobre el programa. La elección perfecta sería:

$$\left(Q(x_t|x_{t-1}^i\right)_{opt} = P(x_t|x_{t-1}^i, z_t)$$

$$\left(Q(x_t|x_{t-1}^i\right)_{opt} = \frac{P(z_t|x_t, x_{t-1}^i)P(x_t|x_{t-1}^i)}{P(z_t|x_{t-1}^i)}$$

Entonces, los pesos dependerían de:

$$w_t^i \propto w_{t-1}^i P(z_t|x_{(t-1)}^i)$$

Sin embargo, para que se cumpla el caso óptimo, tiene que ser posible realizar el muestreo sobre $P(x_t|x_{t-1}^i, z_t)$ y el cálculo en $P(z_t|x_{t-1}^i)$.

**Sequential Importance Resampling (SIR)**

El "*Sequential Importance Resample*" (SIR) es una mejora del algoritmo SIS que añade pasos de remuestreo. El remuestreo significa que las partículas con menor peso tienden a colocarse en las posiciones de partículas con mayor peso, pero una vez recolocadas, se seguirán evaluando por separado. Tras realizar la técnica de remuestreo, se asume que todas las partículas tienen la misma posibilidad de acierto, porque se han agrupado en los casos con más peso, entonces todos los pesos tendrán el mismo valor (Turner, 2013):

$$w_t^i = \frac{1}{np}$$

La principal ventaja de las técnicas de remuestreo en el SIS es la facilidad de implementación, pero este modelo puede desencadenar otro problema denominado *Empobrecimiento de la muestra*. Debido a que en el remuestreo las partículas van a tender a agruparse cerca de los casos con más peso, se va a perder la diversidad en los resultados. En casos extremos, las partículas podrían permanecer en un único punto (Orhan, 2012).

Las técnicas de remuestreo también hacen que los datos recabados en la observación del momento actual pierdan importancia, pues a la hora de remuestrear no estamos

haciendo uso de las mismas, sino que estamos usando cálculos aleatorios para conseguir la nueva muestra (Orhan, 2012).

# Desarrollo del proyecto

Tratando de utilizar de la mejor forma posible los algoritmos de visión por computador, el resultado final estará basado en una serie de experimentos de ensayo-error. Por ello, en el desarrollo siguiente serán comentadas las fases del código indicando sus ventajas, inconvenientes y posibles mejoras.

Para este proyecto será necesario que la imagen adquirida por la cámara vea de la mejor manera posible el fondo de la caja, de modo que las paredes de la misma no influyan produciendo oclusiones indeseadas de los objetos. Para ello, se colocará la cámara Kinect en una posición elevada apuntando hacia la mesa donde se realice el ejercicio. La utilización de un muro o pared para colocar la cámara a la altura indicada sería la opción más recomendada, ya que se necesita que la cámara no sufra movimientos indeseados. No obstante, esto supondrá que el material necesario para realizar el ejercicio sea difícilmente transportable, y por ello se recomienta la colocación de la cámara en una estructura estable de fácil montaje que, aunque incurra en más problemas de captación de resultados si se tambalea, permite que el programa pueda ser transportado a diferentes lugares.

*Figura 20.* Imagen de la estructura utilizada durante el desarrollo

# Detector múltipe de cubos

La primera idea es realizar un filtro que sea capaz de detectar y seguir todos los cubos durante el minuto que dura el experimento. Saber la posición de los cubos en todo momento ayudará a prevenir los problemas de oclusión que ocurren con bastante frecuencia en los ejercicios.

**Detección de caida de objetos**

El filtro de partículas no se puede crear a lo largo de toda la imagen, pues entonces serían necesarias demasiadas partículas para ser capaces de detectar un nuevo objeto. Por lo tanto, utilizando la detección de movimiento se va a calcular la posición final del nuevo cubo en el lado de la caja correspondiente. El filtro de movimiento compara el movimiento surgido entre dos fotogramas seguidos para detectar el momento en el que el cubo deja de moverse. El último punto calculado se considerará como el punto final del viaje del cubo.

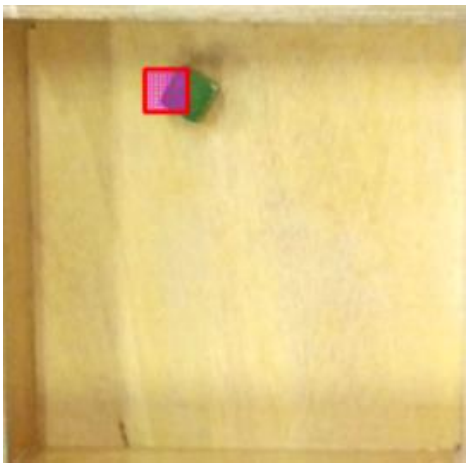


*Figura 21.* Ejemplo de detección del punto de caida del cubo

La figura 20 muestra la zona aproximada donde se encuentra el cubo nuevo. Su centro es el punto calculado anteriormente. Sobre esta zona se realizará el cálculo del color del cubo.

**Cálculo del color con el espacio RGB**

Es imposible realizar matemáticamente el cálculo de la diferencia del color con respecto a los colores objetivos porque la velocidad del cubo lanzado hace complicado detectar un punto del objeto en el que se aprecien bien las características del color. No obstante, una vez el cubo permanece estático, se puede medir con mayor precisión el color. Como se ha explicado anteriormente, el espacio RGB se puede expresar como un cubo, por lo tanto se puede expresar la distancia entre dos colores como si se tratase de la distancia de dos puntos en el plano. Por un lado se tendrán los valores de los colores medidos ($rgb_m$) en la caja y por otro los valores buscados ($rgb_o$) de manera que:

$$Distancia_{rgb} = \sqrt{(r_m - r_o)^2 + (b_m - b_o)^2 + (g_m - g_o)^2}$$

El programa comparará esta distancia con todos los colores que se estén buscando, y devolverá como resultado el color que más se asemeje al medido.

**Filtro de partículas por color**

Es necesario crear un algoritmo que sea capaz de rastrear cubos usando una cantidad pequeña de partículas. Esto beneficiará a que el programa pueda contabilizar una cantidad mayor de cubos antes de tener una carga computacional excesiva. A continuación se explicarán las fases del filtro creado.

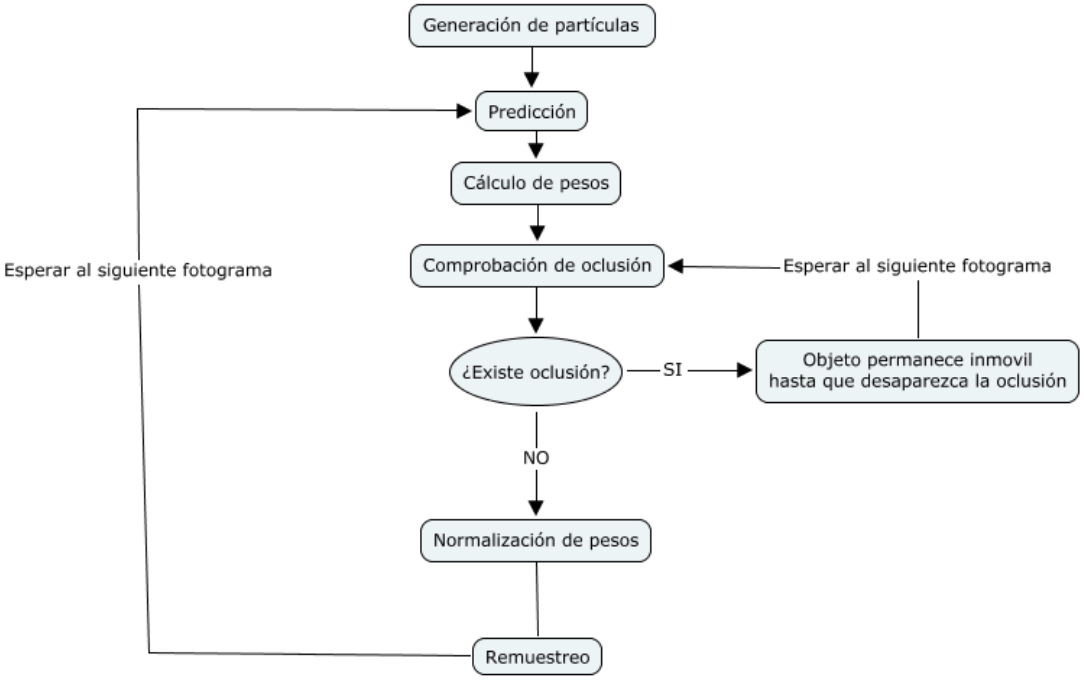



Figura *22*. Orden de las tareas del filtro de partículas por color

***Generación de partículas***

Las partículas se generarán en una pequeña zona alrededor del punto final de movimiento, con el fin de que el filtro creado no aparezca en una zona alejada del objeto. Dentro de esta zona, el posicionamiento de las partículas es aleatorio, mientras que la velocidad inicial se considerará nula.

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_{inicial} = \begin{bmatrix} aleatorio \\ aleatorio \\ 0 \\ 0 \end{bmatrix}$$

***Predicción***

Antes de realizar los cálculos correspondientes sobre la imagen, se predice tanto la posición como la velocidad que tiene el filtro de partículas usando los datos calculados en la anterior iteración del algoritmo. Como no existen datos de aceleración en el problema, la velocidad dependerá unicamente de ella misma en el momento anterior,

pero la posición se calculará con los datos recabados sobre la posición y velocidad. De manera que:

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_{t-1} + \begin{bmatrix} v_x \\ v_y \\ v_{\dot{x}} \\ v_{\dot{y}} \end{bmatrix}$$

donde $\vec{v}$ es un parámetro ajustable de la solución que nos indican las variaciones de posicionamiento y velocidad.

***Cálculo de probabilidad***

Para decidir qué punto calculado tiene mayor peso se usa la siguiente fórmula:

$$Probabilidad = \frac{\exp\left(\frac{-distancia^2}{\sigma^2}\right)}{\sqrt{2\pi\sigma}}$$

donde el valor $\sigma$ determina la sensibilidad del filtro usado. Cuanto mayor sea el valor del parámetro, menos sensible será el algoritmo.

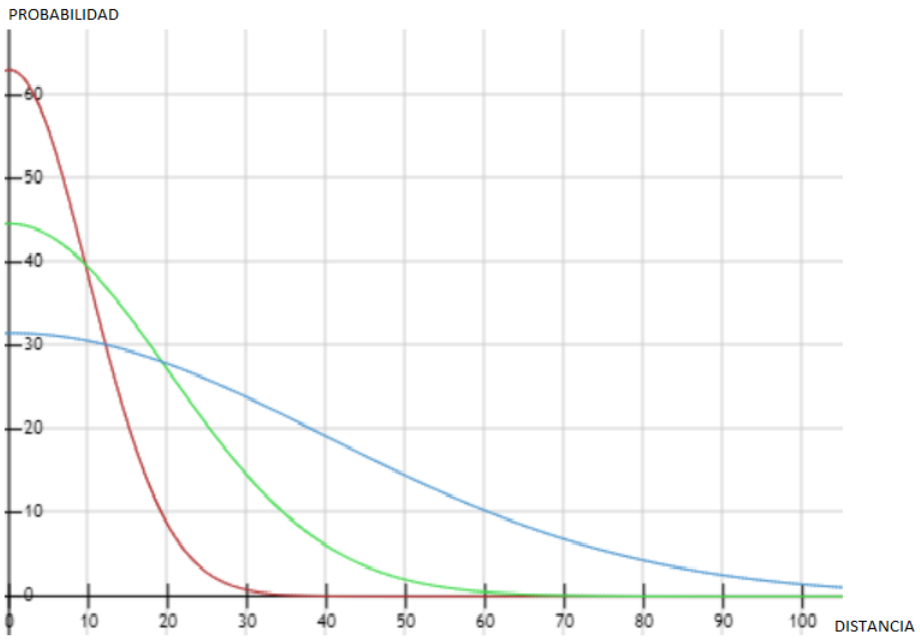



*Figura 23.* Probabilidad según el parametro $\sigma$: Rojo $\sigma$ =10, Verde $\sigma$ =20, Azul $\sigma$ =40

Como se puede apreciar en el gráfico de la figura 22, si usamos un valor de $\sigma$ pequeño, solo los casos cuya distancia de color sea cercana al real serán significativos. Para un valor de sensibilidad mayor, como es la función en azul, muchos de los colores menos cercanos pueden dar una respuesta positiva, así que se debe usar en casos donde el color objetivo no se confunda fácilmente con otros elementos.

Los resultados numéricos de la probabilidad anterior aún no se pueden considerar como los pesos de las partículas. Para ello hay que realizar una etapa de normalización para que la suma de los pesos sea igual a la unidad, por lo tanto los valores máximos y mínimos de la función de probabilidad no influyen en los calculos posteriores.

*Figura 24*. Ejemplo de movimiento de las partículas para la búsqueda de un cubo

Las partículas que se aparecen en la figura 23 muestran la posición de los casos calculados tras realizar la etapa de predicción. Remarcadas en color azul, destacan las partículas con mayor peso.

***Detección de oclusiones***

En muchos casos el cubo puede estar ocluido por otro, lo que conlleva que el filtro de partículas no pueda realizar el cálculo anterior de probabilidad correctamente. Para prevenir esto, cada vez que se detecte correctamente el cubo, éste se asignará a un área

cuadrada de aproximadamente el tamaño del objeto que lo contenga. De esta manera, en futuros cálculos se podrá comprobar si una partícula se encuentra calculando un punto de un área que pertenece a otro objeto. En caso de que esto se repita varias veces a lo largo de todo el listado de partículas, éste se considerará ocluido. En este caso los cálculos sobre las partículas no se volverán a producir hasta comprobar que el cubo se muestra correctamente en pantalla.

***Remuestreo***

Como se observa en la figura 24, una vez se ha calculado el peso de las partículas en función a la probabilidad de acierto (función marcada en rojo), se acabará la iteración remuestreando los casos para que la mayoría de ellas se agrupen en los más posibles.

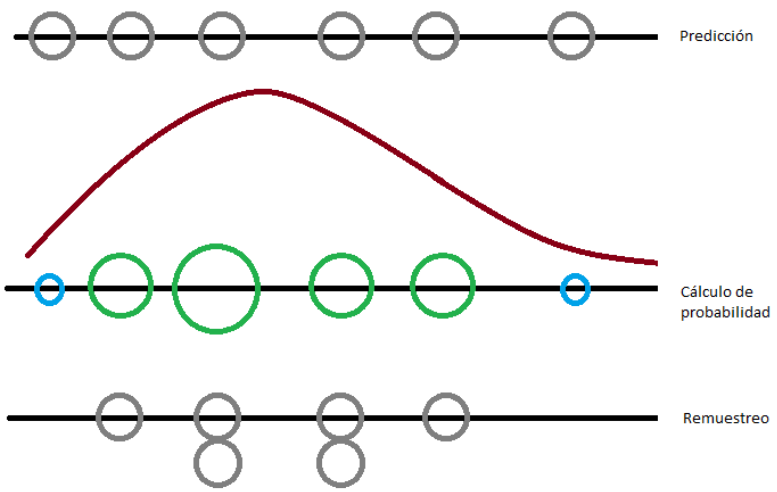



*Figura 25.* Variaciones de las partículas

Que una partícula tenga poco peso no significará que no se remuestreen partículas en su posición, simplemente que será menos probable.

**Conclusiones de la solución propuesta**

Hay que destacar negativamente que no se puede realizar el ejercicio del BBT correctamente con el algoritmo creado, pues detecta movimiento en todo momento para encontrar la posición del cubo nuevo, por lo que pasar la mano al otro lado de la banda

delimitadora otorgará datos de movimientos de la mano, lo que conlleva que la posición final sea errónea. Por tanto, en las pruebas han lanzarse los cubos para que el experimento no se vea alterado por este factor, lo cual es contraproducente para la recuperación del paciente.

Otro punto en contra es la elección de la posición final del nuevo objeto, ya que funciona correctamente mientras que no existan choques entre los cubos. En el momento que el objeto nuevo desplace a uno anterior, el programa los confundirá, haciéndolo bastante impreciso.

Respecto al conteo de cubos, el programa realiza correctamente la detección de los mismos y sus oclusiones, y además es capaz de comparar los datos de filtros activos con el de movimientos detectados para comprobar los aciertos y errores cometidos. Sin embargo, no tiene la habilidad de mantener esta precisión cuando aparecen numerosos objetos por falta de fluidez del vídeo. Esto demuestra que un seguimiento exhaustivo de los objetos requiere un valor computacional demasiado alto para la aplicación propuesta.

Por otro lado, se incrementa el número de errores si la cámara o la caja son movidas, normalmente debido a golpes que pueden recibir al realizar el ejercicio. Esto se debe a que el programa no es capaz de diferenciar entre los tipos de movimientos existentes.

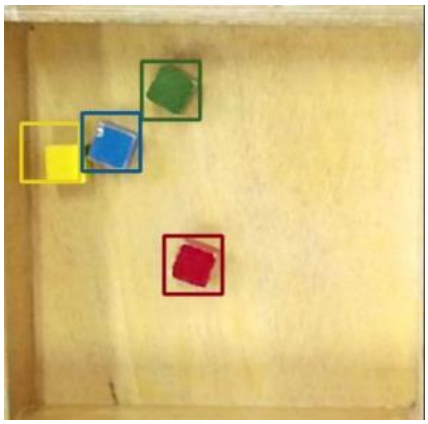


*Figura 26.* Ejemplo de detección de múltiples objetos

El espacio de color RGB demuestra no ser el más indicado para la detección del color, ya que comete muchos errores a la hora de decidir qué color es más correcto. Los casos más significativos son la confusión entre el color azul y el verde, y el conteo excesivo del color amarillo al confundirlo con el color de la caja. Por eso es necesario valorar otros espacios de color distintos.

## Filtro de partículas de detección de movimiento

Para intentar mejorar el programa anterior, se va a eliminar el seguimiento continuo de los cubos, de manera que el programa realice el seguimiento de un único cubo a la vez. Esto evitará el alto valor computacional del caso anterior, pues una vez detectado y contabilizado el cubo, se eliminará el filtro y no se volverá a iniciar otro hasta que se detecta la entrada de un nuevo objeto.

**Sustracción de fondo**

Se comprobará la efectividad de la detección de movimiento por sustracción de fondo a cambio del control de velocidad anterior. Con ello pretendemos tener un mejor control de la imagen una vez los objetos estén estáticos, y así poder señalizar mejor la posición final del cubo. Siempre después de encontrar un objeto, el programa recalculará el fondo para que los cubos antiguos formen parte del mismo y reducir los errores que pueden suponer los choques entre objetos.

**Comprobación del color con los espacio HSV y CIELAB**

Se implementa una variable de elección del espacio de color para que se pueda cambiar fácilmente entre estos tres espacios: RGB, HSV y Lab. Aparte, como la iluminación es un elemento clave en el cálculo de color, encender o apagar una luz puede significar un aumento considerable en los errores del conteo de colores. Para resolver este problema, el programa puede guardar los datos de color buscados desde el menú principal, simplemente será necesario colocar cada cubo en la posición

correspondiente y marcada por el programa. Es conveniente realizar esta tarea siempre después de comenzar la aplicación. En la figura 26 se pueden observar las posiciones de cada cubo. El color del círculo pertenece a los colores de los objetos en CIELAB, en orden descendente: amarillo, rojo, azul y verde.

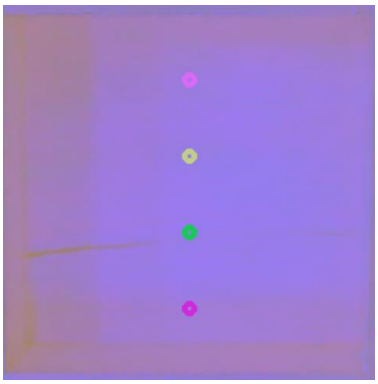


*Figura 27.* Posiciones para la marcación de colores en el espacio CIELAB

Calcular la distancia del color en un espaco cónico como el HSV supone cálculos mucho más complejos que los anteriores. Por ello, el cálculo será realizado dependiendo de la diferencia porcentual de las tres componentes, donde la mayor diferencia del parámetro del tono corresponde a 180 grados de separación con el color medido:

$$Distancia_{hsv} = \sqrt{(h_m - h_o)^2 + (s_m - s_o)^2 + (v_m - v_o)^2}$$

Respecto al espacio CIELAB, Konica Minolta (s.f.) explica el uso de la diferencia Delta E para la medición de distancias entre colores, el cual valora las tres componentes de el espacio L*a*b por igual:

$$\Delta E = \sqrt{(L_m - L_o)^2 + (a_m - a_o)^2 + (b_m - b_o)^2}$$

**Detección de movimiento con partículas**

En este caso, el filtro ya no actuará sobre una imagen a color. En cambio actuará sobre una imagen binaria que corresponde a la sustracción de fondo. Esto conlleva que el filtro de partículas también será binario, es decir, que sólo distinguirá dependiendo de si detecta o no el movimiento. Usar un filtro de partículas en este caso sigue siendo útil

porque a la hora de hacer el seguimiento de la trayectoria del objeto, el algoritmo de predicción puede ayudar a no equivocarlo con otros cubos con los que choca y así tener una idea más acertada de la posición en la que permanece inmóvil.

***Entrada de objetos***

Para evitar movimientos de partículas innecesarios cuando no existen objetos que localizar, el filtro solo se generará en caso de exista movimiento en la zona correspondiente a la tabla delimitadora por donde entran los cubos. También se realizará la eliminación del mismo una vez se ha terminado la trayectoria del cubo.

***Cálculo de probabilidad***

El cálculo de probabilidad ya no necesita de cálculos complejos porque sólo tienen que valorar dos opciones, y por tanto tendrá la siguiente forma:

$$w_i = \begin{cases} 1 & si\ p(x,y)_i = 255 \\ 0 & Cualquier\ otro\ caso \end{cases}$$

donde $p(x,y)$ es el valor del pixel en la imagen umbralizada donde se aplica el algoritmo.

En esta ocasión, la probabilidad se calculará siempre en cada iteración porque se excluye el algoritmo de oclusiones.

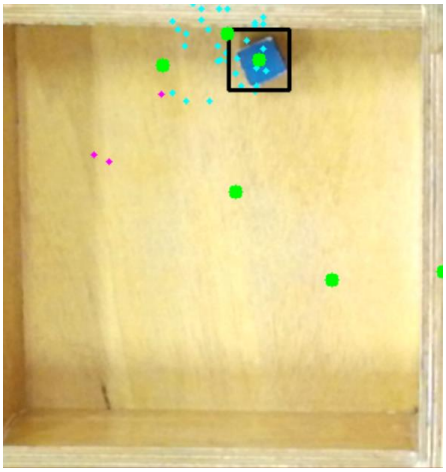


*Figura 28.* Seguimiento de trayectorias del cubo

En la figura 27 se puede ver un ejermplo en el que se realiza el cálculo de trayectoria del nuevo objeto. En ella se aprecia que el filtro es capaz de detectar un cambio brusco de dirección que se puede producir cuando el objeto entra en contacto con la caja.

**Conclusiones de la solución propuesta**

Al experimentar con el código, se puede observar que se resuelve el problema de valor computacional del código. En esta ocasión el programa puede fluir correctamente a tiempo real, y esto permite que el programa contabilice un mayor número de cubos. Experimentalmente puede contar 25 objetos con el 100% de precisión, aunque el algoritmo no permite que se contabilicen más porque, al no ser una localización de múltiples objetos, necesita que el que entra primero sea procesado antes de que entre el siguiente. Como resultado, si varios cubos son lanzados correctamente pero muy seguidos, el programa sólo reconocerá uno de ellos.

Como se puede observar comparando las figuras 20 y 27, el posicionamiento final del cubo se calcula mejor con el método de detección de fondo, pues mientras que en el algoritmo anterior marcaba la zona de caída cerca del borde del cubo, en esta ocasión el cubo aparece perfectamente encuadrado en un área.

No obstante este método, en vez de resolver el problema de choques de cubos, lo empeora. Esto se debe a que cuando un cubo anterior es desplazado, la señal errónea que le otorga al filtro de partículas se mantiene una mayor cantidad de tiempo.

En esta etapa, que usa tres espacios de color, se deben realizar pruebas para comprobar cuál de ellos es más efectivo. Estas pruebas consisten en el lanzamiento de cubos cuando la caja se encuentre vacía, para asegurar que la detección de posición del cubo no sea errónea. Han sido realizadas 20 pruebas por cada tipo de cubo en cada espacio, consiguiendo los siguientes resultados:

Tabla 2.

Porcentaje de acierto del color para cada tipo de cubo en cada espacio usado

| | RGB | HSV | L*a*b |
|---|---|---|---|
| Amarillo | 75% | 95% | 100% |
| Rojo | 95% | 100% | 100% |
| Azul | 70% | 80% | 95% |
| Verde | 80% | 85% | 95% |

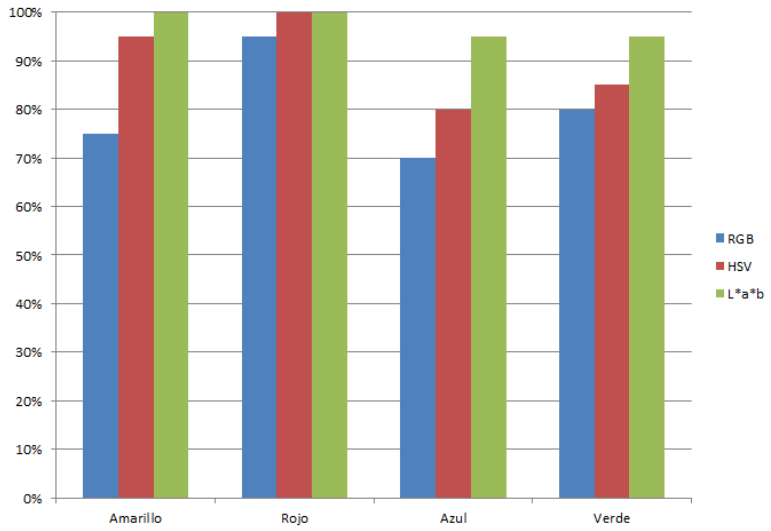



Figura 29. Gráfica de porcentajes de acierto de cada tipo de cubo

El espacio RGB ha demostrado ser el menos útil de los tres. Las sombras que pueden producir las paredes de la caja afectan gravemente a la comprobación del color, con el añadido de no tener la capacidad de hacer una distinción correcta entre el verde y el azul.

En el espacio HSV, aunque aparece una mejora clara con el color amarillo, los cubos verdes y azules son todavía confundidos.

Por último, el CIELAB muestra grandes resultados en la detección de color rojo y amarillo, mientras que aún presenta errores con los otros dos. Por ello, el espacio a utilizar a partir de este momento será el L*a*b, y más adelante podrán ser corregidos los errores ajustando lo máximo posible los parámetros de medida de color.

# Resultado final

A lo largo del desarrollo se ha intentado buscar el punto de la imagen en el cuál se pudiera medir correctamente el color, pero por problemas de choques entre los cubos se comprueba que no es sencillo distinguir los cuerpos antiguos de los nuevos. Mantener un control de la posición de los objetos es una tarea que no se puede realizar a tiempo real por el elevado valor computacional que aparece. Es por esto que se llega a la conclusión de que realizar un seguimiento de la trayectoria del cubo es una tarea que sólo complica el problema. El filtro de partículas usado como resultado final sólo tiene como finalidad la detección de colores y la contabilización de los mismos.

## Control de entrada de objetos y conteo por profundidad

El conteo de cubos totales se realizará independientemente del conteo de cubos de cada color.

Anteriormente la entrada de un objeto ha sido detectada mediante el movimiento, pero este método devuelve una gran cantidad problemas y errores. Como mejora, se utilizará la imagen de profundidad que proporciona la cámara Kinect de manera que sólo detecte los movimientos por encima de la tabla delimitadora. Además, la configuración tratará a los cuerpos pequeños, como por ejemplo los cubos usados, como señales demasiado pequeñas como para que puedan ser contabilizadas como puntos, por lo que se obviarán los cubos simplemente lanzados, solucionando los problemas de ejecución del ejercicio.

Aún existe un caso en el que el conteo no otorga la respuesta adecuada. Si el traslado de cubos no se consigue cumplimentar porque el objeto se escape de las manos o caiga al chocar con la tabla delimitadora, el programa seguirá detectando un objeto si el

paciente ha llegado a acercar la mano a la tabla delimitadora, produciendo un conteo mayor que el real.

# Detección y conteo de colores en movimiento por filtro de partículas

## Segmentación por color

Para observar el movimiento de colores en el programa, se van a realizar umbralizaciones sobre cada uno de ellos de manera que se muestren los datos del movimiento de cada color.

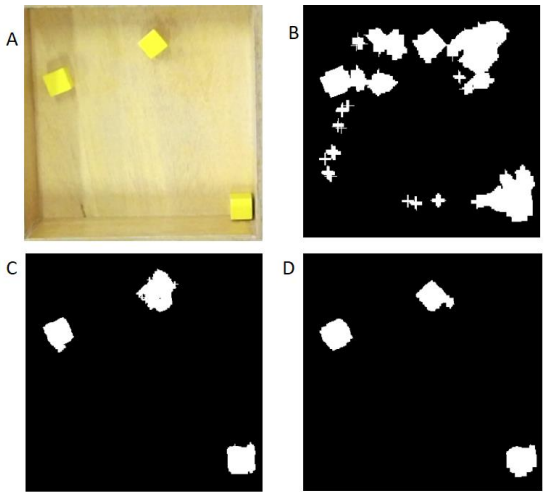



*Figura 30.* Ejemplo de segmentación de colores: A) imagen de la caja. B) Umbralización del color amarillo. C) Sustracción de fondo. D) Segmentación final del color amarillo

En la figura 29 se puede ver un ejemplo de segmentación del color amarillo. Utilizando la imagen de la caja A, se puede realizar una umbralización del color de manera que se muestren los píxeles cuyos colores se encuentren en un rango de colores cercano al color amarillo marcado con anterioridad. Con la utilización de un espacio concreto, el CIELAB, se ajustarán los parámetros de cada color para que el programa no confunda el azul con el verde. El resultado se puede observar en la foto B, donde se

puede apreciar que el color marrón de la caja es muy parecido al cubo y por lo tanto aparecen muchos errores. Si se intenta solucionar el problema disminuyendo la sensibilidad de la umbralización, se corre el riesgo de perder datos de color en el propio cubo, y por lo tanto se opta por la comparación del fotograma con una imagen de sustracción de fondo, imagen C, que detectará todos los objetos no pertenecientes a la caja, sin importar el color. Si las imágenes B y C son combinadas, se obtienen como resultado las posiciones de los cubos en la caja. Posteriormente, se deberán comprobar los movimientos que surgen en la imagen final comparando fotogramas.

**Detección del color**

En realidad, el filtro de partículas no es necesario para una simple detección de movimiento, pero teniendo en cuenta los errores surgidos al encontrarse muchos cubos en la caja, ha de realizarse un corto seguimiento de los objetos siempre que el filtro detecte más de un color en movimiento. Esto puede ocurrir cuando se tarda mucho en reconocer el cubo nuevo y se confunde o con uno anterior que ha sido tapado por la mano, o con otro que se esté moviendo debido a una colisión con el primero.

Se generará un filtro de partículas para cada tipo de cubo en el momento en que se detecte la entrada de un objeto nuevo. La búsqueda se realizará sobre la imagen de movimiento de color correspondiente y, cuando un único filtro otorgue una respuesta positiva, se incrementará el valor del contador de su color. Una vez detectado el color, el programa eliminará los datos de los filtros. No obstante, si las respuestas positivas se repiten en más de un filtro, éste seguira el cuerpo identificado hasta comprobar qué señal es la más duradera, pues en los casos de respuesta múltiple se suelen corregir en fotogramas posteriores, porque se puede suponer que son errores de ruido que no permanecerán durante mucho tiempo. El tiempo de búsqueda se medirá por fotogramas

usados para cada objeto nuevo. Si el filtro revisa 15 fotogramas sin concluir en un solo color, se eliminarán los filtros y se omitirá la contabilización del color del cubo.

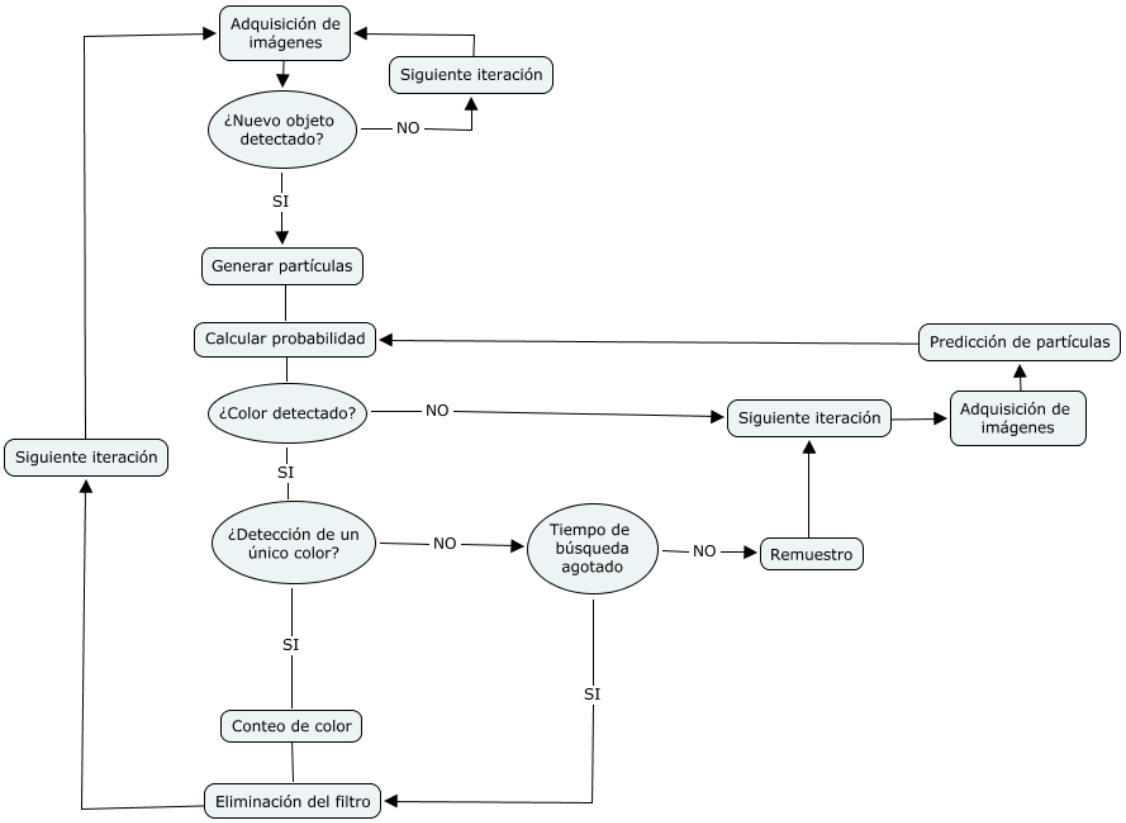



*Figura 31.* Funcionamietno del algoritmo del filtro de partículas múltiple para colores en movimiento

# Otras funciones del código

## Detección de la caja

Durante todo el proceso se han mostrado imágenes del lado de la caja correspondiente al ejercicio, aunque en la imagen extraida por la cámara se podrá observar un área mucho más grande. Encontrar está caja de manera automática ayudará al ejercicio a hacerse más rápido, pues no será necesario colocar la caja manualmente y además el paciente podrá colocar los materiales de la manera que le sea más conveniente para realizar el ejercicio.

Para la detección de la caja será necesario que el lado a buscar se encuentre completamente vacío a la hora de realizar la búsqueda. Usando una imagen de la profundidad umbralizada se pueden diferenciar las figuras que se encuentren a 90 centímetros, que es la distancia a la que se encuentra la cámara Kinect en la estructura usada. En esta imagen se pueden observar las formas cuadradas a la altura de la caja, pese a que se debe realizar una detección de bordes para reconocer más claramente que figuras se pueden observar. De todas las fronteras creadas el programa se centrará en aquellas que tengan 4 vértices diferenciados. Como se ha evitado la aparición de cuadrados del entorno con la umbralización, se considerará que el programa ha encontrado el objeto cuando sólo exista una figura en estas condiciones.

Respecto al resto de figuras de la imagen, el lado opuesto de la caja no se reconocerá como un cuadrado porque los cubos que se encuentran en su interior están variando los contornos de la caja, pues se están buscando figuras cuadradas completas, no sólo los perímetros cuadrados. Si se habla de los cubos, son demasiado pequeños para que el programa los detecte como cuadrados.

**Menú de interacción**

Es importante que el programa se pueda ejecutar fácilmente por cualquier médico o paciente que quiera realizar el ejercicio. Es por ello que se ha creado un menú sencillo que muestra los datos y las funciones del sistema. Desde éste se podrán marcar los cubos, comenzar un ejercicio, elegir la mano a usar en la prueba y ver resultados de ejercicios anteriores. Con ello, aparte de realizar la prueba de manera sencilla y cómoda, también se podrá ver el progreso de los pacientes.

Este menú también mostrará por pantalla la temporización del ejercicio, de manera que cuando acabe el minuto de tiempo que dura la prueba, se dejan de contabilizar los cubos y mandará una señal al programa para guardar los datos del ejercicio.

# Experimentación

La puntuación que se puede conseguir en el experimento variará según la edad y gravedad de la lesión del paciente. Mientras que una persona joven y sana puede llegar a traspasar 70 cubos en un minuto, un paciente que tenga problemas de movilidad del brazo pero haya mejorado tras la rehabilitación apenas podrá conseguir 30 puntos. Para comprobar la eficiencia del programa, se realizarán ejercicios variando la velocidad de lanzamiento de cubos por minutos.

En los experimentos realizados se contabilizan los errores de conteo individuales que surgen en el programa, es decir, en caso de que un movimiento de cubo sea omitido aunque el resultado final sea acertado por errores posteriores, el número de errores individuales será de, como mínimo, dos.

Se realizan series de diez pruebas por cada nivel de velocidad, a evaluar con la intención de detectar el número de veces que realiza mal el conteo de cubos. Adicionalmente, se medirá el porcentaje de las pruebas que realizan una cuenta sin ningún error.

*Tabla 3.*

*Resultados del conteo total de la puntuación del ejercicio BBT*

| Cubos lanzados por minuto | Ciclos completos | Porcentaje de cubos correctamente contados |
|---|---|---|
| 20 | 100,0% | 100,0% |
| 25 | 100,0% | 100,0% |
| 30 | 100,0% | 100,0% |
| 35 | 100,0% | 100,0% |
| 40 | 100,0% | 100,0% |
| 45 | 90,0% | 99,8% |
| 50 | 84,6% | 99,7% |
| 55 | 81,8% | 99,5% |
| 60 | 75,0% | 99,6% |
| 65 | 62,5% | 99,4% |

Gráficamente los resultados se muestran de la siguiente manera:

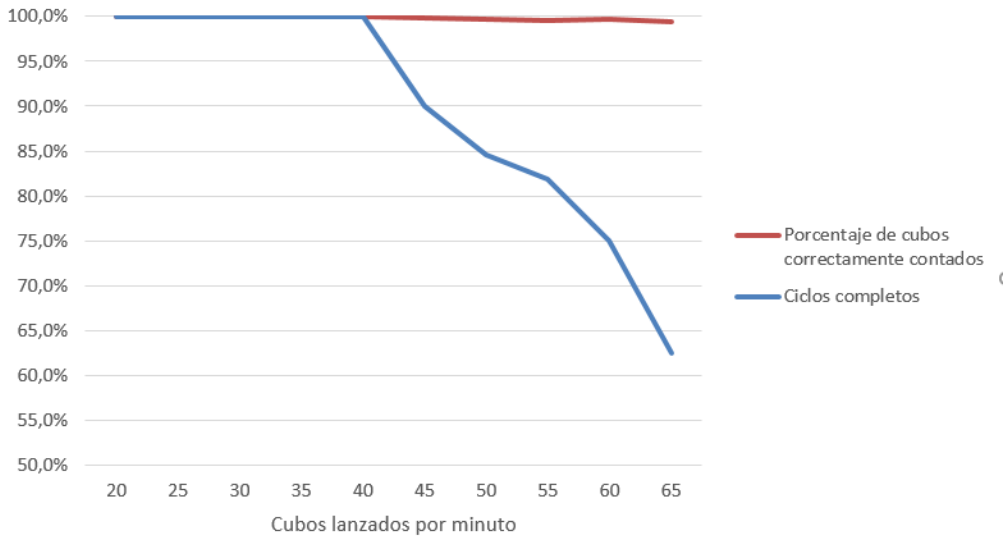



*Figura 32*. Gráfica de resultados del conteo de la puntuación del ejercicio BBT

Se observa que los errrores de conteo aparecen cuando la cantidad de cubos traspasados por minuto es mayor, y esto se debe a que la velocidad de realización del experimento hace que la persona que lo está realizando cometa más errores, como por ejemplo perder los cubos antes de poder realizar el movimiento completo. De todas formas, no es normal que estos errores se repitan muchas veces a lo largo de un ejercicio, y es por eso que la gráfica de porcentaje de cubos contados correctamente no se ve afectada gravemente a lo largo de los experimentos.

El interés de usar el conteo de los colores es observar si algún cubo en especial afecta negativamente al conteo general de alguna manera. Por el color, el objeto puede realizar una cuenta errónea mientras que el resto funciona con normalidad. Pero el conteo general de cubos es mucho más eficaz que el conteo por colores, por lo que no se puede hacer una comparación correcta entre ellos.

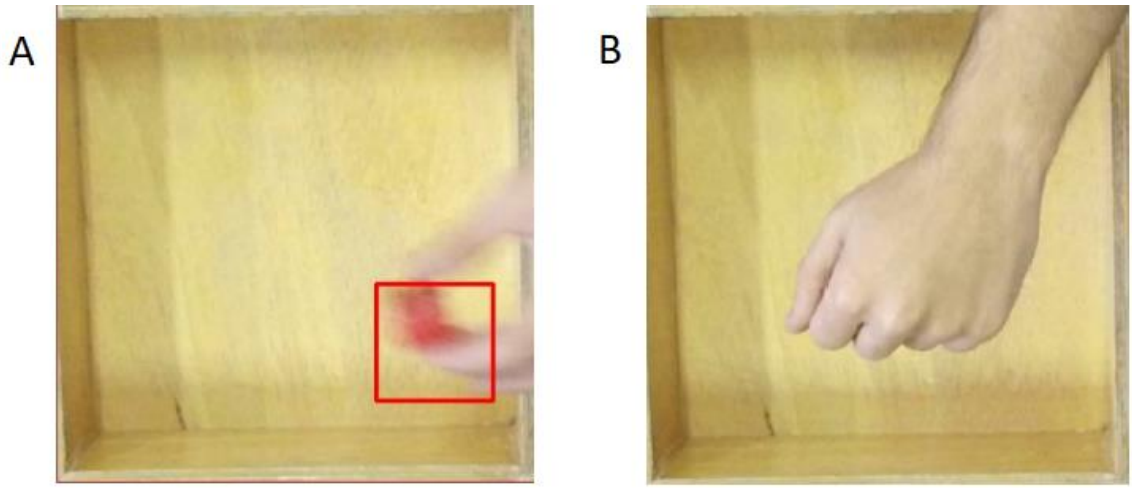



*Figura 33*. Métodos de traspaso de objetos utilizado para el conteo de color

Por otro lado, para que el conteo de colores sea más preciso, el cubo tiene que ser lo más visible posible para la cámara, suceso poco probable ya que el movimiento más cómodo para realizar el ejercicio no es mostrando el objeto a cámara. Para comprobar la diferencia de eficacia según la manera de lanzar el objeto, se realizará otra serie de experimentos en los cuales se utilizarán los dos métodos que se pueden ver en la figura 32: mostrando el objeto al entrar como en la imagen A, o tapándolo completamente hasta completar el movimiento como en la imagen B.

*Tabla 4.*

*Porcentaje de aciertos del color del cubo en el ejercicio BBT*

| Cubos lanzados por minuto | Traspaso de cubos ocultando el objeto | Traspaso de cubos mostrando el objeto |
|---|---|---|
| 20 | 72,0% | 97,5% |
| 25 | 48,0% | 88,8% |
| 30 | 47,5% | 88,7% |
| 35 | 61,1% | 87,9% |
| 40 | 55,5% | 93,3% |
| 45 | 52,6% | 87,4% |
| 50 | 58,5% | 85,5% |
| 55 | 77,3% | 87,3% |

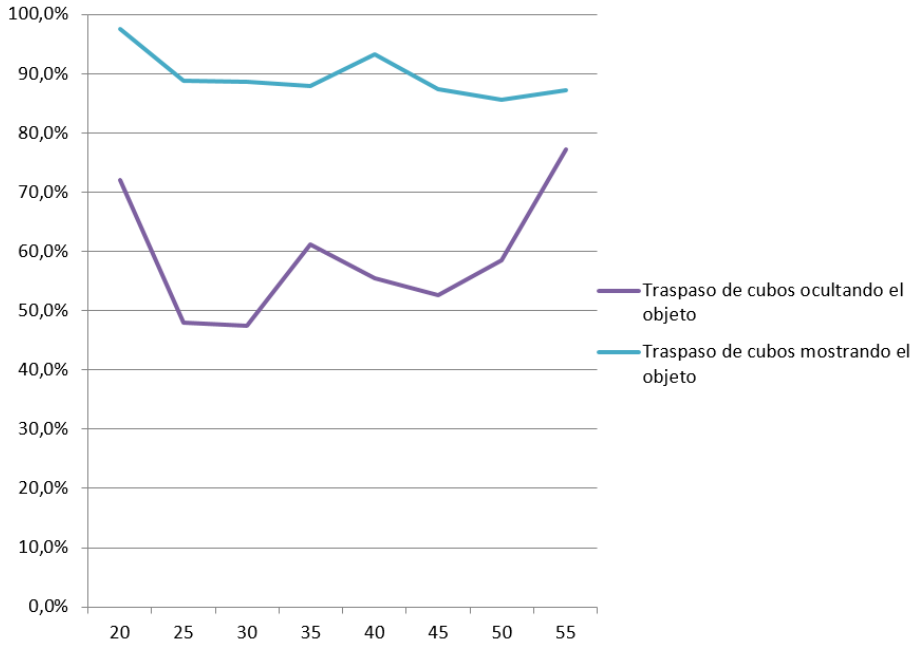



*Figura 34.* Gráfica de porcentaje de acierto del color del cubo en el ejercicio BBT

Los resultados demuestran que, como era lógico, las pruebas realizadas mostrando el cubo incurren en muchos menos errores. No obstante, se debe recalcar que cuando se aumenta considerablemente la velocidad de lanzamiendo de cubos, el segundo método se vuelve considerablemente más preciso. Esto es debido a que lanzar los cubos con más velocidad también influye a que el objeto nuevo aparezca antes en la imagen y, en consecuencia, la mano deje de ser un obstáculo antes.

# Conclusiones

Durante la realización del proyecto, se ha comprobado que el uso del filtro de partículas para la localización y seguimiento múltiple de objetos en movimiento es útil y práctico. Sin embargo, el gasto computacional aumenta según va aumentando el número de cubos desplazados, llegando al punto de que el equipo no pueda procesar tanta información. Como resultado se ha preferido utilizar un filtro más simple que realice simplemente la detecctión del cubo, pero sin realizar seguimiento, lo que permite procesar rápidamente todos los objetos.

Asimismo, se considera necesaria la implantación de otro algoritmo de detección de colores más eficaz con dos objetivos: detectar si algún color en especial tiene más problemas que otros al ser detectado, y hacer una comparativa de este contador con el conteo general de cubos para obtener un resultado más acertado de la puntuación total conseguida en el ejercicio.

Una mayor velocidad de desplazamiento, los cambios en las condiciones de ilumniación del entorno, las oclusiones y desplazamientos por golpes entre cubos, etc., Incrementan la dificultad de la detección y elevan los requerimientos de cómputo.

Ha resaltado también que las funciones utilizadas de visión artificial no consiguen que se contabilicen al 100% todos los objetos, especialmente cuando aumenta la cantidad de puntos conseguidos. El peor casos para la detección se da cuando el usuario realiza el test con la mano menos afectada, que realiza un ejercicio más rápido.

Se ha constatado también que la aplicación no se ve afectada por el movimiento de la cámara, y por tanto sería recomendable la utilización de una estructura fácil de transportar. Esto ayudará a que se pueda realizar la prueba en distintos lugares, sin necesidad de tener reservado un espacio fijo para la misma. Para que los resultados sean

los correctos, la estructura debe tener 90 centímetros de altura, ya que al utilizar la imagen de la profundidad estamos específicando la distancia entre la caja y la cámara.

Durante la investigación para la realización del proyecto, se ha verificado también que el espacio de color CIELAB es el más adecuado para los resultados que necesitan obtenerse ya que, frente a RGB y al HSV, el primero muestra una mayor destreza en la captación y diferenciación de colores debido a que se ve menos afectado por los cambios de iluminación en la caja, lo cuál es indispensable para un correcto conteo de puntos por color.

# Bibliografía

Ashley, J. (5 de marzo de 2014). Quick reference: Kinect 1 vs Kinect 2. Recuperado el 17 de septiembre de 2017, de http://www.imaginativeuniversal.com/blog/2014/03/05/Quick-Reference-Kinect-1-vs-Kinect-2/

de la Escalera, A. (2001). *Visión por Computador. Fundamentos y métodos*. Madrid: Pearson Educación S.A.

Díaz Abreu, L. M. (2015). Estudio del filtro de partículas y de la evolución diferencial aplicados al problema de seguimiento de objetos en video. Trabajo de diploma para obtener el grado de Maestro de Ciencias con Espacialidad en CIancia de la Computación y matemáticas industriales, Centro de Inventigación en Matemáticas A.C., Guanajuato, Mexico. Recuperado el 27 de agosto de https://cimat.repositorioinstitucional.mx/jspui/bitstream/1008/379/2/TE%20550.pdf

García Mateos, G., Jaén Terrones, S., Escarabajal Henarejos, D., Molina Martinez, J., y Hernández Hernández, J. (s.f.). *Segmentación automática de imágenes de cultivos: estudio comparativo de modelos de color*. Recuperado el 20 de septiembre de 2017, de: http://www.interempresas.net/Horticola/Articulos/122034-Segmentacion-automatica-de-imagenes-de-cultivos-estudio-comparativo-de-modelos-de-color.html

González Cid, Y. (s.f.). Aplicaciones de la Ingeniería Electrónica II. Imágenes en color: Espacios de color. Recuperado el 17 de septiembre de 2017, de http://dmi.uib.es/~ygonzalez/Master_10212/Espacios_color_10210.pdf

Greenan, S., Buxton, S., Sillo, O., y Thomas, E. (s.f.). *Box and Block Test*. Recuperado el 13 de septiembre de 2017, de Physiopedia: https://www.physio-pedia.com/Box_and_Block_Test

Grzonka, S., Grisetti, G., y Burgard, W. (12 de mayo de 2009). Towards a Navigation System for Autonomous Indoor Flying. Freiburg, Alemania. Recuperado el 25 de septeimbre de 2017, de http://www.slawomir.de/publications/grzonka09icra/grzonka09icra.pdf

Hounslow, K. (27 de enero de 2014). OpenCV Tutorial: Real-Time Object Tracking Without Colour [video]. Vancouver, Canada. Recuperado el 25 de agosto de 2017, de https://www.youtube.com/watch?v=X6rPdRZzgjg

Illana, J. I. (26 de enero de 2013). Métodos Monte Carlo. Universidad de Granada, España. Recuperado el 27 de agosto de 2017, de http://www.ugr.es/~jillana/Docencia/FM/mc.pdf

Kari, P., Baksheev, A., Kornyakov, K., y Eruhimov, V. (22 de abril de 2012). Realtime Computer Vision with OpenCV [Versión electrónica]. *ACM Queue*, *10*(4). Recuperado el 28 de agosto de 2017, de http://queue.acm.org/detail.cfm?id=2206309

Konica Minolta. (s.f.). Entendiendo El Espacio de Color CIE L*A*B*. Recuperado el 18 de septiembre de 2017, de http://sensing.konicaminolta.com.mx/2014/09/entendiendo-el-espacio-de-color-cie-lab/

La Ley Orgánica 1/1982, de 5 de mayo, de *Protección civil del derecho al honor, a la intimidad personal y familiar y a la propia imagen* [Versión electrónica]. Boletín Oficial del Estado, 115, de 14 de mayo de 1982. Recuperado el 26 de septiembre de 2017, de https://www.boe.es/buscar/act.php?id=BOE-A-2002-22188

la Ley Orgánica 15/1999, de 13 de diciembre, de *Protección de Datos de Carácter*

*Personal* [Versión electrónica]. Boletín Oficial del Estado, 298, de 14 de diciembre de 1999. Recuperado el 26 de septiembre de 2017, de https://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750

Ley Orgánica 41/2002, de 14 de noviembre, de *básica reguladora de la autonomía del paciente y de derechos y obligaciones en materia de información y documentación clínica* [Versión electrónica]. Boletín Oficial del Estado, 15 de noviembre de 2002. Recuperado el 26 de septiembre de 2017, de https://www.boe.es/buscar/act.php?id=BOE-A-2002-22188

Maggio, E., y Cavallaro, A. (2011). *Video Tracking. Theory and practice.* Chinchester, Reino Unido: John Wiley & Sons Ltd.

Mathiowetz, V., Federman, S., y Wiemer, D. (diciembre de 1985). Box and Blocks Test of Manual Dexterity Norms for 6-19 Years Olds [Versión electrónica]. *Canadian Journal of Occupational Therapy*, *52*(5). Recuperado el 23 de agosto de 2017, de https://www.researchgate.net/profile/Virgil_Mathiowetz/publication/270718227_Box_and_Block_Test_of_Manual_Dexterity_Norms_for_6-19_Year_Olds/links/55ff2b6308aec948c4f9b06d/Box-and-Block-Test-of-Manual-Dexterity-Norms-for-6-19-Year-Olds.pdf

McGee, L. A., y Schmidt, S. F. (noviembre de 1985). Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry [Versión electrónica]. *NASA Technical Memorandum*. Recuperado el 26 de agosto de 2017, de https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19860003843.pdf

McNames, J. (16 de abril de 2012). Particle Filters [Video]. Portland, Oregón, Estados Unidos. Recuperado el 28 de agosto de 2017, de https://www.youtube.com/watch?v=1ACOCUglpbE

Monge, S. (24 de junio de 2011). Software de reconocimiento de emociones a través expresiones faciales. Recuperado el 20 de septiembre de 2017, de http://neuromarca.com/blog/reconocimiento-facial-emociones/

OpenCV. (s.f.). Basic Thresholding Operations. Recuperado el 20 de septiembre de 2017, de http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html

OpenCV. (s.f.). *OpenCV*. Recuperado el 23 de agosto de 2017, de http://opencv.org: http://opencv.org/about.html

Orhan, E. (11 de agosto de 2012). Particle Filtering. University of Rochester, Nueva York, Estados Unidos. Recuperado el 28 de agosto de 2017, de http://corevision.cns.nyu.edu/~eorhan/notes/particle-filtering.pdf

Platero, C. (2017). Apuntes de Visión Artificial. Universidad Politécnica de Madrid, España. Recuperado el 31 de agosto de 2017, de http://www.elai.upm.es/moodle/mod/resource/view.php?id=452

Quinoces Riesco, A. (26 de enero de 2016). España, puntera en robótica asistencial dirigida a ancianos y enfermos. Recuperado el 26 de septiembre de 2017, de http://www.efefuturo.com/noticia/espana-puntera-en-robotica-asistencial-dirigida-a-ancianos-y-enfermos/

Rebello, S. (29 de octubre de 2009). Tracking motion capture. Digital Studio Me. Recuperado el 29 de agosto de 2017, de http://www.digitalstudiome.com/article-1936-tracking-motion-capture/

Turner, L. (13 de mayo de 2013). An Introduction to Partivle Filtering. Recuperado el 27 de agosto de 2017, de http://www.lancaster.ac.uk/pg/turnerl/PartileFiltering.pdf

Vaas, L. (19 de febrero de 2015). *Avance en el reconocimiento facial: el 'detector de caras Dense Profunda'.* Recuperado el 21 de septiembre de 2017, de https://nakedsecurity.sophos.com/es/2015/02/19/breakthrough-in-facial-recognition-the-deep-dense-face-detector/

Vezzani, R., Baltieri, D., y Cucchiara, R. (2015). *Pathnodes integration of standalone Particle Filters for people tracking on distributed surveillance systems.* Recuperado el 25 de septiembre de 2017, de http://imagelab.ing.unimore.it/imagelab2015/pubblicazioni/vezzani_ICIAP09.pdf

Xataka Windows. (7 de octubre de 2013). La evolución de Kinect y la importancia real de Microsoft Research. Recuperado el 25 de agosto de 2017, de https://www.xatakawindows.com/xbox/la-evolucion-de-kinect-y-la-importancia-de-microsoft-research

# Anexo A: Planificación de proyecto

En este anexo serán enumerados el número de tareas necesarias para la realizacion del proyecto, y se especificará su duración. La planificación mostrada se corresponde con la duración aproximada de cada una de las tareas realizadas durante la experimentación exceptuando las pruebas con los pacientes, que no se han llegado a realizar. Al final, se muestra un diagrama de Gantt con la planificación del proyecto óptima.

**A) Documentación y familiarización con las herramientas y los materiales**

Deben conocerse las distintas las funciones que proporciona el software utilizado en el desarrollo, especialmente las librerías de visión OpenCV y Kinect para Windows, independientemente del lenguaje de programación que se plantee utilizar, ya sea Phyton, C, C++, etc. En este apartado también se incluye el tiempo necesario para integrar estos programas en el ordenador. Será también conveniente conocer el algoritmo del filtros "*Bootstrap*".

Por otra parte, es importante tener en cuenta las características de los elementos en la BBT, como la altura de la estructura o las instrucciones del ejercicio. Aunque esta etapa pueda llevarse a cabo en un espacio corto de tiempo, debe conocerse bien esta información. La comprensión de estos datos supone una gran ventaja en el desarrollo del proyecto.

**B) Implantación del código**

Debe crearse un código que cumpla todas y cada una de las cualidades expresadas anteriormente en el proyecto: detección automática de la caja, detección automática de los colores, conteo general de la puntuación, conteo por colores de la puntuación y menú de interacción con el médico o paciente.

**C) Testeo y corrección de errores**

Antes realizar las pruebas reales con pacientes, debe llevarse a cabo una etapa de prueba en el lugar de investigación para buscar y corregir los errores de programación y ejecución que pudiesen surgir en el programa.

**D) Primeras pruebas con pacientes**

Se realizarán pruebas reales con pacientes, ya sea en el lugar de trabajo o transportando al estructura al lugar donde se encuentren las personas que se van a beneficiar del programa. Las pruebas se harán siempre bajo la supervisión de un equipo médico cualificado.

**E) Ajuste de parámetros en el programa tras las pruebas**

Tras las pruebas serán valorados los parámetros del programa que han de variarse. Los datos recabados informarán de si es necesario alterar los ajustes de los algoritmos de conteo para que sean más eficaces, o si los integrantes de la prueba han conseguido entender y usar el programa sin dificultad.

**F) Pruebas finales con pacientes**

Tras haber realizado los cambios pertinentes, se realizarán unas segundas pruebas con los pacientes. Los resultados de éstas indicarán la eficacia del programa final. Se podrá valorar tras esta etapa si el programa necesita algún cambio o prueba adicional con el fin de conseguir una mejora significante.

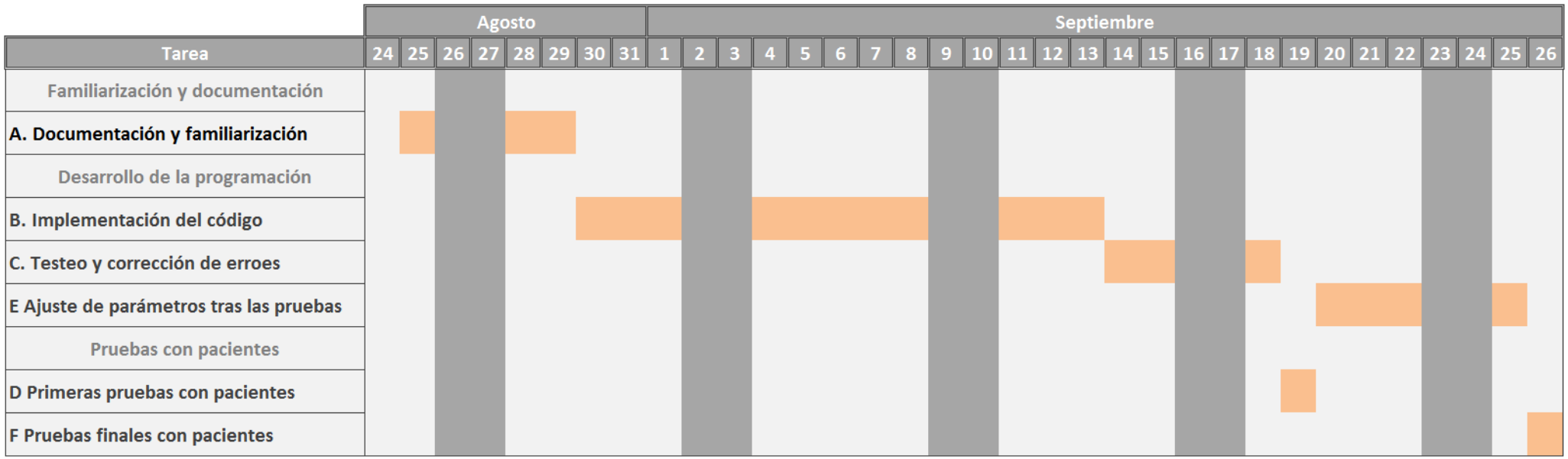
*Tabla Anexo A- 1.*

*Duración de fechas aproximadas de las tareas*

| Tarea | Fecha de inicio | Duración (h) | Fecha de finalización |
|---|---|---|---|
| A | 25/08/2017 | 32 | 30/08/2017 |
| B | 31/08/2017 | 80 | 13/09/2017 |
| C | 14/09/2017 | 24 | 18/09/2017 |
| D | 19/09/2017 | 4 | 19/09/2017 |
| E | 20/09/2017 | 32 | 25/09/2017 |
| F | 26/09/2017 | 4 | 26/09/2017 |

*Tabla Anexo A- 2.*

*Diagrama de Gantt de la planificación del proyecto*



# Anexo B: Marco Socio-Económico

La robótica asistencial en España se encuentra en una etapa de auge, principalmente en lo que se refiere al cuidado de ancianos y enfermos. El potencial de este campo en la sociedad digital actual es enorme, aunque aún se puede mejorar en gran medida la capacidad de percepción del entorno de los equipos y su interacción con los humanos. (Quincoces Riesco, 2016). Sin embargo, la aportación económica necesaria para investigación y desarrollo de estas aplicaciones normalmente es muy elevada, en especial cuando se utilizan máquinas complejas, como los robots humanoides. Por ello, los proyectos con una aplicación real en centros dedicacos a la salud deben ser lo más económicos posible para que se implemente a mayor escala.

A continuación, serán valorados los gastos económicos que supone la realización de este proyecto. Estos gastos incluyen las herramientas para el desarrollo del proyecto, las horas de trabajo empleadas por el ingeniero y otros gastos indirectos.

- **Costes materiales:** incluyen todas las herramientas y equipos completamente necesarios. En este apartado de valoran también los programas informáticos necesarios y elementos necesarios para realizar el transporte de todo el conjunto al lugar de pruebas con pacientes.

- **Costes de desarrollo:** en este apartado serán valoradas las horas trabajadas por el ingeniero para documentarse, crear y comprobar el funcionamiento del programa tanto en el lugar de trabajo como fuera realizando pruebas con pacientes.

- **Costes indirectos:** son los costes que, sin estar ligados directamente al proyecto, son necesarios para el correcto desarrollo del mismo. En estos gastos se incluyen

elementos como la conexión a internet, la luz y el transporte de los materiales. Los costes indirectos se supondrán, aproximadamente, el 5% del total de los gastos.

Tabla Anexo B- 1.

Costes materiales

| Item | Unidades | Precio unitario | Precio total |
|---|---|---|---|
| **Costes materiales** | | | **2.317,01 €** |
| Material ejercicio BBT | 1 | 348,03 € | 348,03 € |
| Estructura | 1 | 50,00 € | 50,00 € |
| Ordenador | 1 | 1.100,00 € | 1.100,00 € |
| Cámara Kinect 2.0 | 1 | 144,99 € | 144,99 € |
| Kinect para Windows | 1 | 99,99 € | 99,99 € |
| Librería OpenCV | 1 | Gratuito | |
| Software Visual Studio | 1 | 574,00 € | 574,00 € |

*Tabla Anexo B- 2.*

*Costes de desarrollo*

| Item | Unidades | Precio unitario | Precio total |
|---|---|---|---|
| **Costes de desarrollo** | | | **2.680,00 €** |
| Documentación | 32 | 15,00 € | 480,00 € |
| Programación | 80 | 15,00 € | 1.200,00 € |
| Corrección de errores | 56 | 15,00 € | 840,00 € |
| Pruebas fuera de oficina | 8 | 20,00 € | 160,00 € |

Tabla Anexo B- 3.

Costes totales

| Item | Unidades | Precio unitario | Precio total |
|---|---|---|---|
| **Costes materiales** | | | **2.317,01 €** |
| **Costes de desarrollo** | | | **2.680,00 €** |
| **Gastos totales** | | | **4.997,01 €** |
| **Costes indirectos** | **5%** | | **249,85 €** |
| | | **Costes totales** | **5.246,86 €** |

# Anexo C: Marco Legal

En este anexo se valoran los distintos aspectos legales que tiene la realización de las pruebas con pacientes reales. Básicamente se resume en dos apartados importantes: las regulaciones sanitarias y las leyes respecto a la captación y posterior uso de las imágenes.

Las regulaciones en la sanidad (Ley Orgánica 41/2002, de 14 de noviembre, de *básica reguladora de la autonomía del paciente y de derechos y obligaciones en materia de información y documentación clínica.* Boletín Oficial del Estado, 15 de noviembre de 2002) establecen que "Los pacientes tienen derecho a conocer, con motivo de cualquier actuación en el ámbito de su salud, toda la información disponible sobre la misma, salvando los supuestos exceptuados por la Ley". Entonces, el equipo médico debe informar correctamente de la utilidad que tiene la prueba BBT en su programa de rehabilitación, así como los posibles riesgos que pueden aparecer durante el test.

La Ley Orgánica 1/1982, de 5 de mayo, de *Protección civil del derecho al honor, a la intimidad personal y familiar y a la propia imagen* (Boletín Oficial del Estado, 115, de 14 de mayo de 1982) nos indica que para poder tomar una imagen de una persona, es necesario su consentimiento. Por lo tanto, el paciente que pretenda realizar el ejercicio automático BBT deberá dar su permiso, expreso e inequívoco, para ser grabado en vídeo durante el tiempo que dure la prueba.

Los datos recabados en el ejercicio se consideran información personal de cada paciente. Es por ello que, siguiendo las regulaciones de la Ley Orgánica 15/1999, de 13 de diciembre, de *Protección de Datos de Carácter Personal (*Boletín Oficial del Estado, 298, de 14 de diciembre de 1999), el afectado debe de ser informado previamente, de forma clara y concisa, de los datos que recoge el ejercicio propuesto. Esta ley permite el

procesamiento de todos los datos personales relativos a la salud del paciente. No obstante, los profesionales a cargo de la prueba deben mantener el secreto profesional y no difundir los resultados. Existen don excepciones: La primera es contar con la aprobación para su distribución y la segunda es que la comunicación de los datos a un tercero tenga como finalidad el cumplimiento de funciones directamente ligadas al procesamiento de los mismos.